Einführung für neue Nutzer
Wing IDE Professional

Wingware
www.wingware.com

Version 2.1.0
April 17, 2006

Wing IDE ist eine leistungsfähige Software-Entwicklungsumgebung für Python. Mit Wing verringert sich die Zeit des Code Schreibens und des Debuggens, so dass Sie qualitativen Code schneller erstellen können.

- Eine geführte Tour finden Sie in **unserem Tutorial**.
- Probieren Sie Wing selbst aus mit unserer **Schnellstart-Anleitung**.
- Wing IDE 1.1 Nutzer können **lernen, wie sie auf Wing IDE 2.0 umstellen**.
- Außerdem verfügbar: **Was spricht für die Verwendung von Wing IDE?**

Inhalt

Copyright (c) 1999-2005 by Wingware. Alle Rechte vorbehalten.:


```
Wingware
P.O. Box 1937
Brookline, MA 02446
United States of America
```

Wing IDE Tutorial

Dieses Dokument stellt Ihnen Wing IDE vor, indem es Sie anhand eines kleinen Code-Beispiels durch sein Funktionsset führt. Eine schnellere, aber weniger informative Einführung bietet die **Wing IDE Schnellstart-Anleitung**.

Zum Beginnen, klicken Sie auf das Symbol `Nächste` (zweites von rechts in Werkzeugleiste genau über dieser Seite).

1.1. Tutorial: So Starten Sie

Zusätzlich zur Installation von Wing IDE müssen Sie auch Python installieren. Dieses Tutorial funktioniert mit Python-Version 2.0 oder höher.

Python können Sie entweder von python.org oder wingware.com herunterladen.

Wenn die obigen Links nicht funktionieren oder den falschen Browser aufschlagen, müssen Sie wahrscheinlich die `BROWSER` Umgebungsvariable auf den Namen der Browser-Executable, die Sie verwenden möchten, setzen (zum Beispiel: `mozilla`) und Wing IDE neu starten.

In Linux/Unix können Sie außerdem eine Browser-Befehlszeile zu Ihrer Einstellung **Befehle der URL-Anzeige** hinzufügen. Dies ist nur empfehlenswert, wenn Ihr bevorzugter Browser bei Bestimmung mit der `BROWSER` Umgebungsvariable nicht funktioniert. Die Einstellung von `BROWSER` wird im Allgemeinen bei der Wiederverwendung von Browser-Instanzen sowie der Erstellung und dem Aufschlagen von Browser-Fenstern (wie benötigt) besser funktionieren.

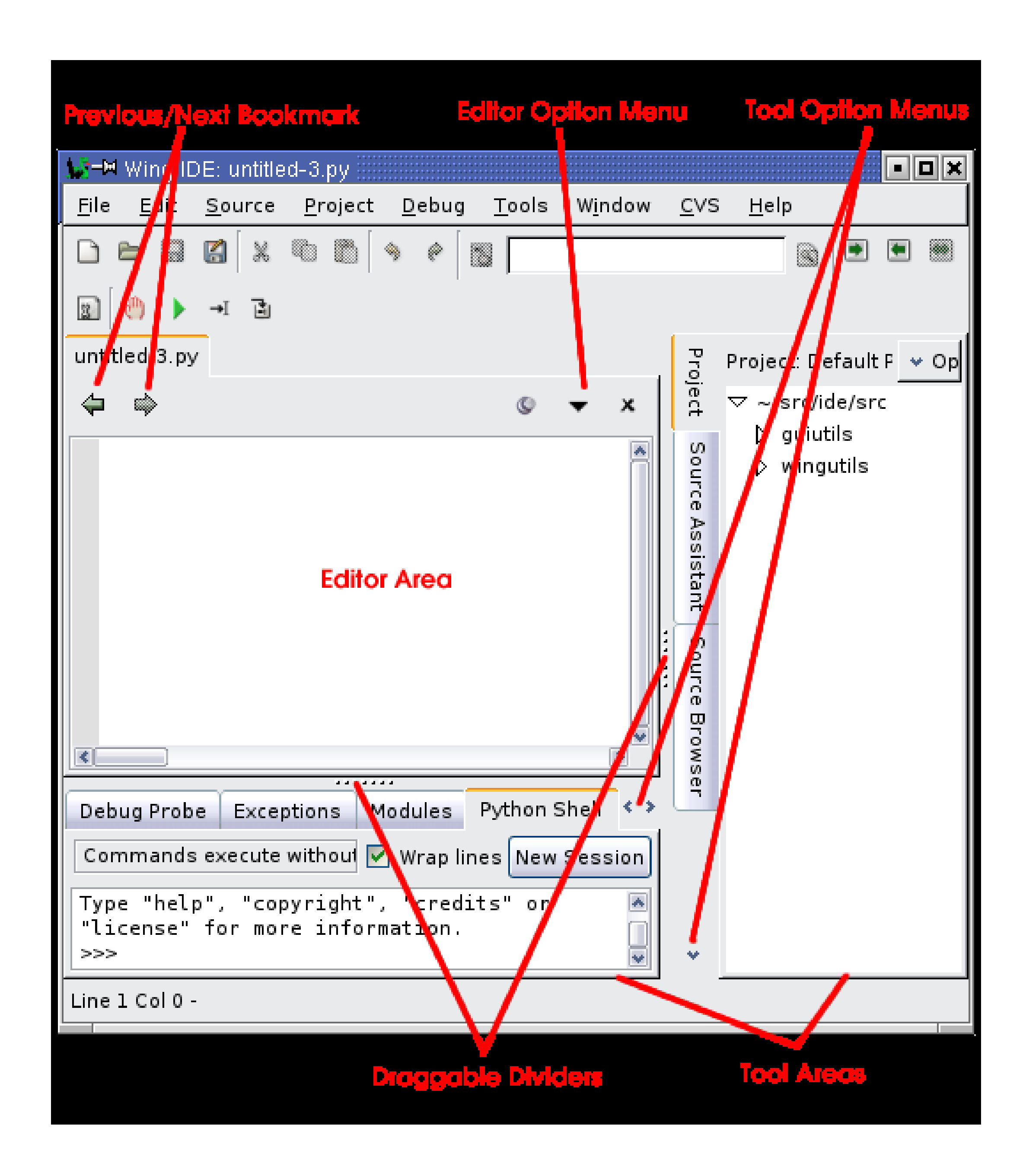
Als nächstes kopieren Sie das gesamte `tutorial` Verzeichnis aus Ihrer Wing IDE Installation an einen Ort, an dem Sie Schreibzugriff auf die Dateien im Verzeichnis haben. Sie können dies manuell vornehmen oder den folgenden Link zur Ausführung eines Skriptes verwenden, das Sie nach einem Zielverzeichnis für das Kopieren der Tutorial-Informationen fragen wird: **Tutorial jetzt kopieren**

Wir sind für Feedback und Fehlerberichte sehr dankbar. Beide können direkt von Wing IDE eingereicht werden, indem Sie die Einträge `Feedback einreichen` und `Fehlerbericht einreichen` aus dem Menü Hilfe verwenden oder uns eine E-Mail an support at wingware.com senden.

1.2. Tutorial: Umgang mit Wing IDE

Lassen Sie uns mit ein paar Grundlagen beginnen, die Ihnen während der Arbeit mit diesem Tutorial den Umgang mit Wing IDE erleichtern.

Die Benutzeroberfläche von Wing IDE ist in einen Editor-Bereich und zwei Werkzeugboxen geteilt, die durch verschiebbare Teilungslinien voneinander getrennt sind. Verwenden Sie die Optionsmenüs in jedem Bereich, um Teilungen anzulegen oder Werkzeuge zu verschieben. Die Schaltflächen Vorheriges/Nächstes Lesezeichen und die Einträge **`Nächstes Dokument`**, **`Vorheriges Dokument`** und **`Letztes Dokument`** im Fenstermenü können verwendet werden, um schnell zwischen den Dokumenten des Editor-Bereiches zu wechseln, wie zum Beispiel zwischen diesem Tutorial und den Source-Dateien, mit denen Sie später arbeiten werden.



Konfigurationsoptionen

Es gibt viele Konfigurationsoptionen, die für die Anpassung der Benutzeroberfläche zur Verfügung stehen. Einige von diesen sind unten beschrieben. Sobald Sie Änderungen an diesen vornehmen, werden die Einstellungen in Ihrer Projetdatei und Ihren Projekteinstellungen gespeichert.

- **Felder teilen** -- Der Editor-Bereich und die Werkzeugboxen können in mehrere

Unterfelder geteilt werden, indem Sie die Optionsmenüs des Editors und der Werkzeugboxen verwenden. Diese Menüs können durch einen Klick auf das Drop-Down-Symbol oder durch einen rechten Mausklick auf die Notizbuchreiter aufgeschlagen werden. Beachten Sie, dass bei einer Teilung des Editor-Bereiches jeder neue Teil die gleichen Dateien anzeigen wird, wie die anderen Felder. Dies ermöglicht Ihnen das Bearbeiten mehrerer Teile der gleichen Datei.

Die Teilung Ihres Editor-Bereiches oder das Anlegen eines separaten `Hilfe`

Werkzeugfensters kann den Umgang mit diesem Tutorial erleichtern.

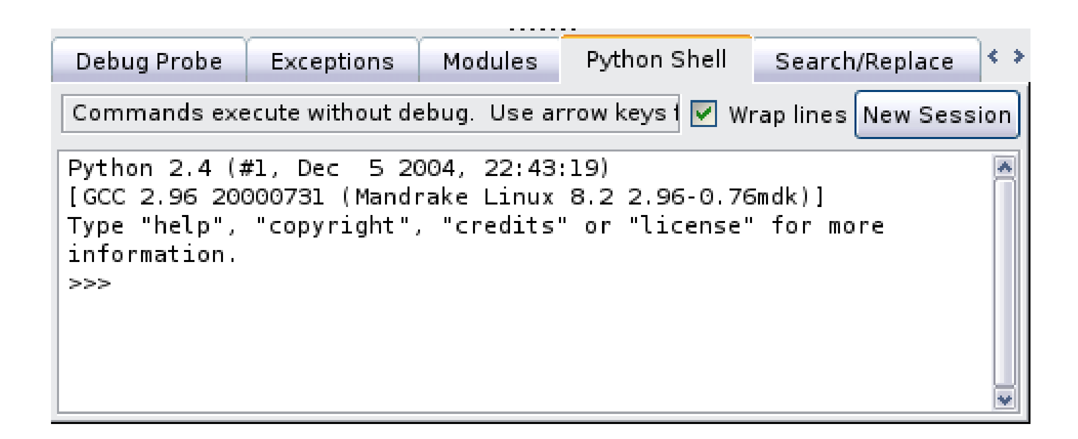
Die Anzahl der Teilungen, die standardmäßig in den Werkzeugboxen angezeigt werden, variiert in Abhängigkeit von der Größe des Monitor.

- **Werkzeugboxen minimieren** -- Ein Klick auf einen bereits aktiven Werkzeugreiter in einer der Werkzeugboxen wird den gesamten Bereich minimieren, so dass nur noch die Reiter für diesen Bereich sichtbar sind. Ein weiterer Klick auf einen beliebigen Reiter wird die ursprüngliche Größe der Werkzeugbox wiederherstellen. Sie können auch F1 und F2 verwenden, um den Status der zwei Werkzeugboxen zu wechseln. Dies ist eine praktische Möglichkeit, um den verfügbaren Raum für den Editor oder die andere Werkzeugbox zu vergrößern.
- **Werkzeuge verschieben und hinzufügen** -- Mit dem Optionsmenü der Werkzeugbox können Werkzeuge zwischen den Werkzeugboxen oder in separate Fenster verschoben werden. Zusätzliche Felder eines Werkzeuges können mit dem Optionsmenü der Werkzeugbox erstellt werden und das Menü `Fenster` erlaubt das Hinzufügen eines Werkzeuges in einem separaten Fenster.
- **Dokumentfenster hinzufügen** -- Zusätzliche Dokumentfenster können auch mit dem Menü `Fenster` erstellt werden. Jedes separate Dokumentfenster enthält sein eigenes Set geöffneter Dateien.
- **Emacs-Individualität** -- Wenn Sie ein Emacs-Nutzer sind, dann können Sie Wing jetzt mit der Einstellung **Individualität** in den Tastaturmodus von Emacs setzen. Vergessen Sie nicht, auf `OK` oder `Anwenden` zu klicken, damit Ihre Änderungen wirksam werden.
- **Andere Optionen** -- Die **Schriftart/-größe des Source-Codes** und die **Schriftart/-größe der Anzeige** können verändert werden. Das Aussehen der Werkzeugleiste kann mit den Einstellungen **Größe der Werkzeugleiste** und **Stil der Werkzeugleiste** geändert werden. Die Werkzeugboxen können von **rechts nach links** oder von **unten nach oben** verschoben werden. Das Optionsmenü des Editors lässt Sie zwischen der Verwendung von Notizbuchreitern und eines Popup-Menüs auswählen, um zwischen geöffneten Editoren zu navigieren.

Zusätzliche Informationen über die Anpassung der Benutzeroberfläche an Ihre Bedürfnisse finden Sie im Kapitel **Anpassung** des Benutzerhandbuches.

1.3. Tutorial: Überprüfen Sie Ihre Integration von Python

Bevor wir mit dem ersten Code beginnen, sollten wir sicherstellen, dass Wing Ihre Python-Installation gefunden hat (die neuste Version wird bevorzugt, wenn Sie mehrere Versionen installiert haben). Öffnen Sie das Werkzeug **Python-Shell**, um dies zu überprüfen. Nach einem kurzen Augenblick sollte es die Python-Befehlsaufforderungszeile wie in dieser Abbildung anzeigen:



Wenn dies nicht funktioniert oder die falsche Python-Version verwendet wird, dann können Sie Wing mit der Einstellung `Python-Executable` in den **Projekteigenschaften**, welche über die Werkzeugleiste oder das Menü `Projekt` verfügbar sind, auf die richtige Version zeigen. Nach dem Ändern dieser Eigenschaft müssen Sie im Werkzeug Python-Shell die Schaltfläche `Neue Sitzung` anklicken.

Sobald die Shell funktioniert, fügen Sie diese Zeilen Python-Code ein, entweder mit Kopieren/Einfügen oder Ziehen und Ablegen:

```
for i in range(0, 10):
  print ' ' * (10 - i) + '*' * i
```

Die sollte ein Dreieck drucken, wie folgt:

Debug Probe | Exceptions | Modules | Python Shell | Search/Replace

Commands execute without debug. Use arrow keys f | Wrap lines | New Session

```
Python 2.4 (#1, Dec  5 2004, 22:43:19)
[GCC 2.96 20000731 (Mandrake Linux 8.2 2.96-0.76mdk)]
Type "help", "copyright", "credits" or "license" for more
information.
>>> for i in range(0, 10):
...     print ' ' * (10 - i) + '*' * i
...

         *
        **
       ***
      ****
     *****
    ******
   *******
  ********
 *********
>>>
```

Beachten Sie, dass die Shell führende Leerräume entfernt, wenn Code-Blöcke in sie kopiert werden. Dies ist hilfreich, wenn Sie Code aus Source-Dateien austesten.

Sie können beliebig viele Fenster des Python-Shell-Werkzeuges erstellen; jedes führt seinen eigenen privaten Prozess aus, der vollständig separat von Wing IDE und Ihrem Debug-Prozess gehalten wird.

1.4. Tutorial: Ein Projekt einrichten

Jetzt sind wir bereit, mit dem Schreiben von Code zu beginnen. Der erste Schritt beim Arbeiten mit Wing IDE besteht darin, eine Projektdatei einzurichten, so dass Wing Ihren Source-Code finden und analysieren kann und damit es Ihre Arbeit über die Sitzungen hinweg speichert.

Wing startet am Anfang mit einem leeren Projekt. Wenn Sie bereits ein Projekt geöffnet haben, wählen Sie jetzt `Neues Projekt` aus dem Menü `Projekt`.

Fügen Sie als nächstes Ihre Source-Dateien zum Projekt hinzu. Sie können dies mit den Einträgen **`Hinzufügen`** aus dem **`Projektmenü`** oder mit einem rechten Mausklick auf das Werkzeug **Projekt** vornehmen. Für die Zwecke des Tutorials verwenden Sie **`Verzeichnisbaum hinzufügen`**, um alle Dateien aus Ihrer Kopie des `tutorial` Verzeichnisses hinzuzufügen.

Nachdem Sie die Dateien hinzugefügt haben, speichern Sie das Projekt auf dem Laufwerk

mit dem Menü `Projekt`. Verwenden Sie `tutorial.wpr` als Namen für die Projektdatei und speichern diese in Ihrem `tutorial` Verzeichnis.

Dateien durchsuchen

Dateien in Ihrem Projekt können mit einem Doppelklick oder einem rechten Mausklick auf die Dateiliste geöffnet werden. Wenn im Menü `Optionen` das Kontrollkästchen `Auswahl folgen` in der oberen rechten Ecke markiert ist, wird Wing für Dateien, die einmal angeklickt werden, auch den Source-Code anzeigen. Diese Dateien werden jedoch im „nicht-sticky" Modus geöffnet, d.h. sie werden automatisch wieder geschlossen, wenn eine andere Datei aufgeschlagen wird. Dies wird durch das Stick-Pin-Symbol in der oberen rechten Ecke des Editor-Bereiches angezeigt:

-- Zeigt an, dass die Datei dauerhaft geöffnet ist, bis sie vom Benutzer explizit geschlossen wird.

-- Zeigt an, dass die Datei vorübergehend geöffnet ist und automatisch geschlossen wird, außer wenn sie bearbeitet wird.

Das Klicken auf das Pin-Symbol wechselt zwischen den Modi, so lange wie die Datei keine ungespeicherten Änderungen enthält. Ein rechter Mausklick auf das Symbol zeigt eine Liste der zuletzt besuchten Dateien an. Beachten Sie, dass diese Liste sowohl vorübergehende als auch dauerhafte Dateien umfasst, während die Liste `Letzte` im Menü `Datei` nur die dauerhaften Dateien enthält.

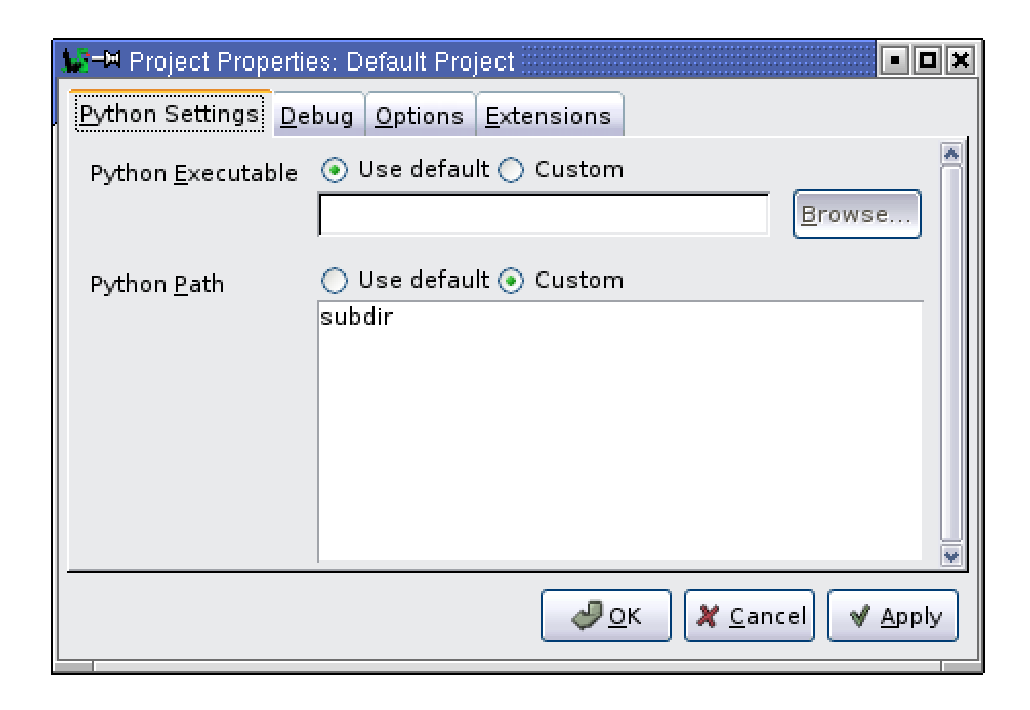
Die Anzahl der vorübergehenden Editoren, die zusätzlich zu den sichtbaren geöffnet bleiben können, wird mit der Einstellung **Schwelle für vorübergehende Dateien** festgelegt.

Dieser Mechanismus verhindert, dass sehr viele Dateien geöffnet werden, wenn Sie im Debugger durch den Code schreiten oder den Source-Browser, Suchmanager und andere Werkzeuge verwenden, um Dateien zu durchsuchen.

Sie können die Projektanzeige ändern, um die Dateien in eine tiefe Hierarchie, eine abgeflachte Hierachie oder nach Mime-Typen zu sortieren. Diese Auswahlmöglichkeiten stehen über das Menü `Optionen` in der Projektansicht zur Verfügung.

PYTHONPATH konfigurieren

Immer wenn Ihre Python-Source von `PYTHONPATH` abhängt (entweder extern gesetzt oder durch die interne Änderung von `sys.path`), müssen Sie auch Wing über Ihren Pfad informieren. Dieser Wert kann im Dialog `Projekteigenschaften`, der über das Menü `Projekt` oder die Werkzeugleiste zugänglich ist, eingegeben werden.



Für dieses Tutorial benötigen Sie mindestens einen `PYTHONPATH`, der das `subdir` Unterverzeichnis Ihres `tutorial` Verzeichnisses einschließt, wie in der obigen Abbildung dargestellt. Dies enthält ein Modul, das als Teil des ersten Coding-Beispiels genutzt wird.

Die Konfiguration wird hier für Illustrationszwecke verwendet. Sie könnten den Beispiel-Code problemslos ohne einen `PYTHONPATH` ausführen, indem Sie die `path_example.py` Datei an die gleiche Stelle verschieben wie das Beispiel-Skript oder indem Sie sie in das „site-packages“ Verzeichnis Ihrer Python-Installation platzieren. Beide Möglichkeiten erlauben Python, die Module ohne geänderten `PYTHONPATH` zu finden.

Gemeinsam genutzte Projektdateien

Wenn Sie beabsichtigen, Wing IDE in einem Entwicklungsteam zu verwenden, das Projektdateien in einem Revisionskontrollsystem, wie `CVS`, `Subversion` oder `Perforce SCM`, gemeinsam nutzt, dann versichern Sie sich, dass Sie Ihr Projekt mit der Eigenschaft

Projektart auf `Gemeinsam` ändern. Dies trennt das Projekt in zwei Dateien: `*.wpr` mit gemeinsam genutzten Projektdaten und `*.wpu` mit nutzerspezifischen Daten. Tragen Sie nur die `*.wpr` Datei in die Revisionskontrolle ein, um Revisionskonflikte, die durch gleichzeitiges Bearbeiten verursacht werden, zu vermeiden.

1.5. Tutorial: Auto-Vervollständiger und Source-Assistent

Wing hat jetzt die Tutorial-Beispiele und alle Module, die importiert und von diesen genutzt werden, gefunden und analysiert. Der Analyseprozess wird im Hintergrund ausgeführt und ermöglicht, dass Wing Ihnen besseren Support während der Prüfung und Bearbeitung von Code bereitstellt. Bei einer größeren Code-Basis können Sie die CPU-Belastung dieses Prozesses wahrnehmen, aber in dem Beispiel des Tutorials wird diese Analyse unmittelbar nach der Konfiguration des Projektes erfolgen.

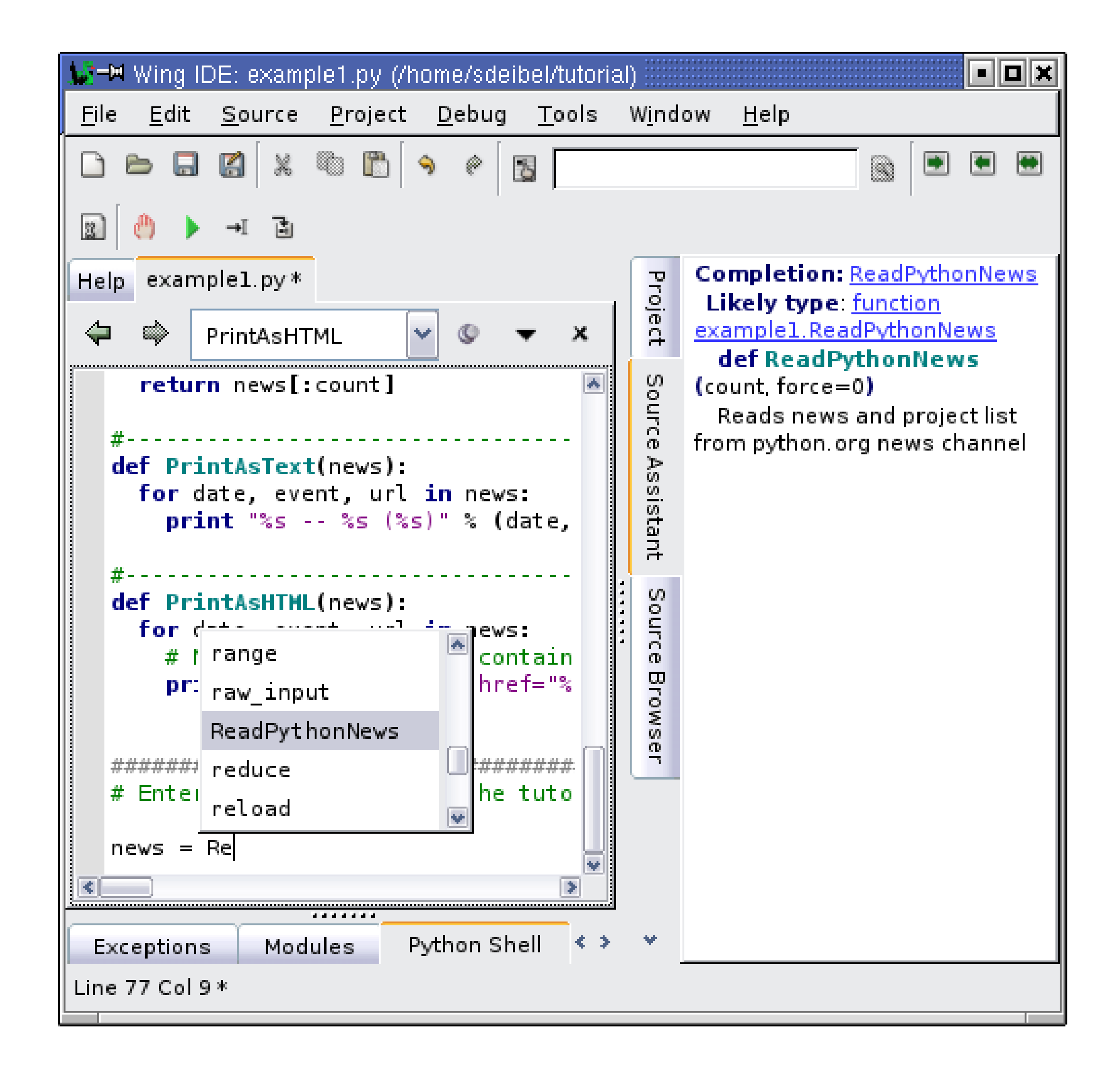
Der Auto-Vervollständiger des Editors und der Source-Assistent sind zwei der wichtigsten analysegesteuerten Werkzeuge in Wing IDE.

Doppelklicken oder rechtsklicken Sie auf die Datei `example1.py` im Projektfeld, um diese Werkzeuge auszuprobieren. Bringen Sie außerdem das Werkzeug **Source-Assistent** nach vorn. Dort zeigt Wing IDE Dokumentation, die Call-Signatur und andere Informationen an, während Sie sich durch Ihren Source-Code bewegen oder in anderen Werkzeugen arbeiten. Es ist also eine gute Idee, dieses Feld die meiste Zeit sichtbar zu lassen.

Rollen Sie bis zum Ende von `example1.py` und tippen den folgenden Code in die Datei ein (nicht einfügen):

```
news = Re
```

Während Sie tippen wird Wing ein Popup-Menü mit Vervollständigungsoptionen aufschlagen. Sie können die Tab-Taste drücken, um den gegenwärtig markierten Wert einzugeben oder mit den Pfeiltasten durch die Liste rollen. Als Sie „news“ eingegeben haben, war dieser Vervollständiger nicht sehr nützlich, da Sie `news` noch nicht als ein Symbol in Ihrem Source-Code definiert hatten. Sobald Sie jedoch mit dem Eingeben von “ = Re“ fortfahren, wird Wing eine weitere Vervollständigungsliste anzeigen, in der `ReadPythonNews` markiert ist. Sie werden bemerken, dass der Source-Assistent aktualisiert wird, um Call-Informationen für diese Funktion oder für den jeweils im Auto-Vervollständiger markierten Wert anzuzeigen:



Drücken Sie als nächstes die `Tab`-Taste, um die Vervollständigung für `ReadPythonNews` einzufügen, und geben dann `(` ein. Sie sollten jetzt diesen Code in Ihrem Editor haben:

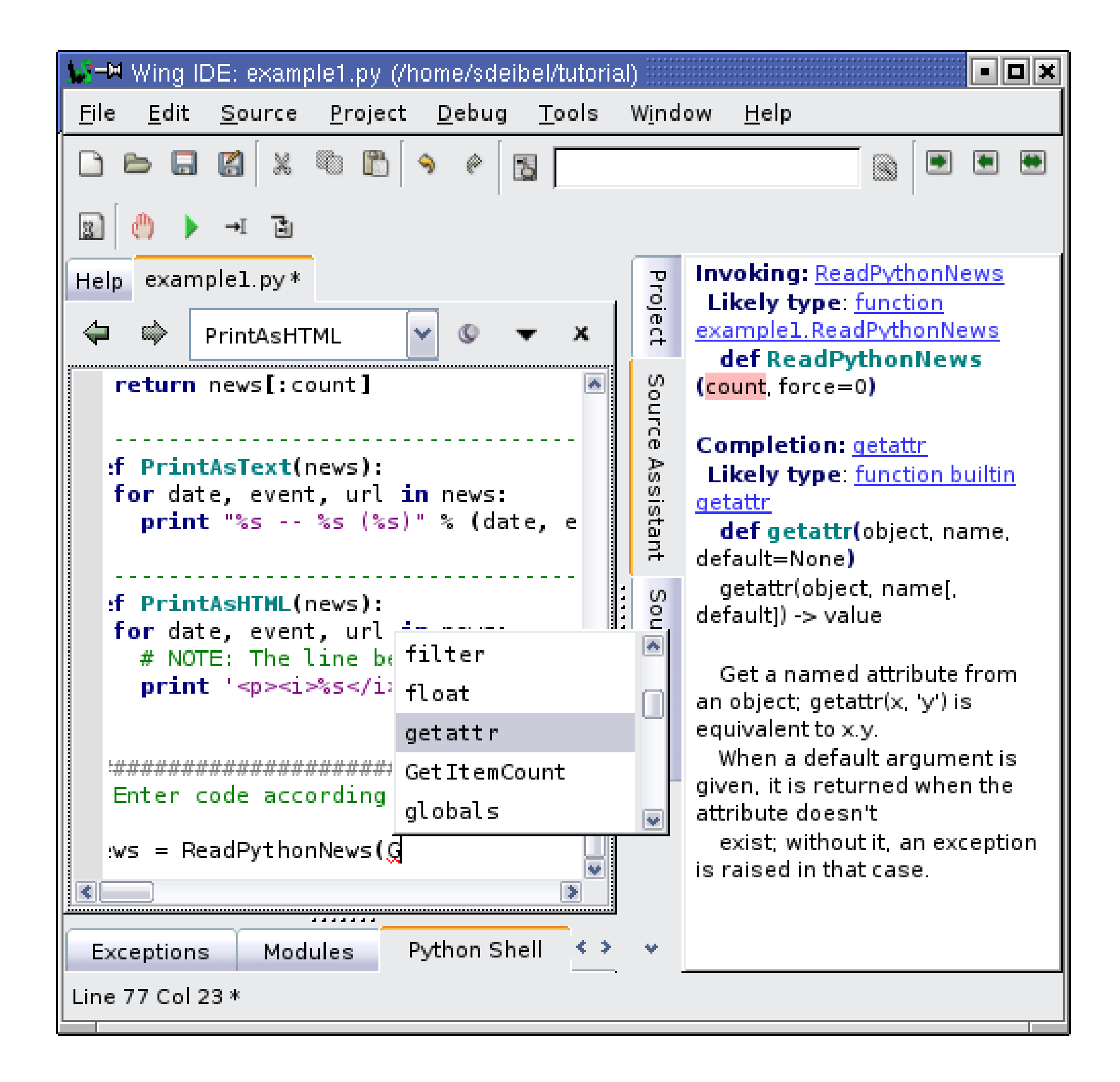
```
news = ReadPythonNews(
```

Wenn Sie es gewöhnt sind, die `Enter`-Taste für die Auto-Vervollständigung zu nutzen,

fügen Sie dies zu der Einstellung **Vervollständigungstasten** hinzu.

Duplicate substitution definition name: „note“.

Drücken Sie `G`, um die Eingabe des ersten Argumentes für `ReadPythonNews` zu beginnen. Sie werden sehen, dass der Source-Assistent seine Anzeige ändert, um das erste Argument in den Call-Informationen für `ReadPythonNews` hervorzuheben und dass er Informationen zu dem Vervollständigungswert des Argumentes hinzufügt:



Der Doc-String für `ReadPythonNews` wird vorübergehend versteckt, um Platz auf dem Bildschirm zu schaffen (aber dies kann mit der Option `Doc-String während der Vervollständigung anzeigen` geändert werden. Sie erreichen diesen Eintrag über das Kontextmenü, das mit einem rechten Mausklick auf die Oberfläche des Source-Assistenten aufgeschlagen wird.)

Fahren Sie jetzt mit der Eingabe der restlichen Source-Zeile fort, so dass die folgende, fast vollständige Zeile mit Source-Code vorliegt (das Zeichen `)` am Ende fehlt):

```
news = ReadPythonNews(GetItemCount()
```

Drücken Sie ein paar Mal die Enter-Taste. Sie werden bemerken, dass Wing IDE die nachfolgenden Zeilen automatisch einrückt und rote Fehlermarkierungen unter ihnen anzeigt, kurz nachdem Sie die Eingabe beendet haben. Dies zeigt an, dass es in Ihrem Code einen Syntax-Fehler gibt.

```
Help  example1.py*
PrintAsHTML

def PrintAsText(news):
  for date, event, url in news:
    print "%s -- %s (%s)" % (date, event, url)

#-----------------------------------------------------------------
def PrintAsHTML(news):
  for date, event, url in news:
    # NOTE: The line below contains a deliberate typo
    print '<p><i>%s</i> <a href="%s">%s</a></p>' % (data, url,

#################################################################
# Enter code according to the tutorial here:

news = ReadPythonNews(GetItemCount()
```

Sobald Sie die Zeile korrigieren und Sie mit dem `)` Zeichen vervollständigen, werden die Fehlermarkierungen gelöscht. Sie sollten jetzt diese vollständige Code-Zeile in Ihrer Datei haben:

```
news = ReadPythonNews(GetItemCount())
```

Der Source-Assistent wird auch aktualisiert, während Sie den Cursor in Ihrem Editor bewegen. Versuchen Sie zum Beispiel in `GetItemCount` zu gehen. Beachten Sie auch, dass die blauen Links im Source-Assistenten verwendet werden können, um zum Punkt der Definition von jedem Symbol, das dort aufgelistet ist, zu springen. Bei Variablen springt der Link nach `Symbol:` zum Punkt der Definition dieser Variablen, während die Links nach `Wahrscheinliche Art:` zum Punkt der Definition dieses Datentyps gehen (diese sind die gleichen, wenn das Symbol eine Funktion, Methode oder Klasse ist; wir werden den Source-Assistenten mit interessanterem Code später ausprobieren).

Geben Sie die folgenden zusätzlichen Code-Zeilen ein, um ein bisschen mehr mit diesen Werkzeugen zu spielen:

```
PrintAsText(news)
PrintAsHTML(news)
```

Jetzt haben Sie ein vollständiges Programm, das im Debugger ausgeführt werden kann. Es gibt viele weitere Editor-Funktionen, die es wert sind zu lernen. Wir werden später in diesem Tutorial auf sie zurückkommen.

1.6. Tutorial: Debuggen

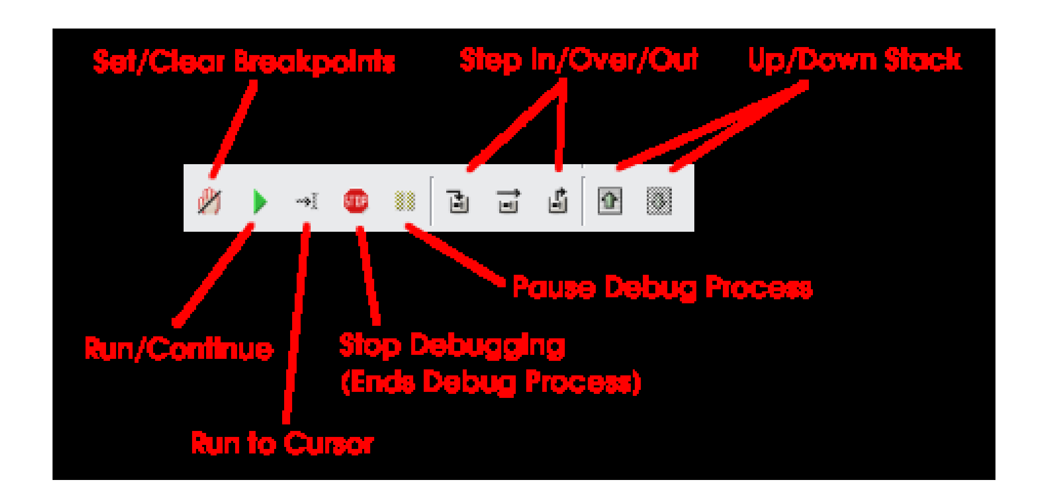
Für den Fall, dass Sie es noch nicht herausgefunden haben: Das `example1.py` Programm, das Sie erstellt haben, verbindet via HTTP zu `python.org`, liest und verarbeitet die mit Python verbundenen News-Einträge, die dort bereitgestellt sind, und druckt dann die letzten fünf Einträge als Text und HTML. Machen Sie sich keine Sorgen, wenn Sie auf Ihrer Maschine keine Internet-Verbindung haben; das Skript hat Daten hinterlegt, die verwendet werden, wenn es nicht zu `python.org` verbinden kann.

Zum Starten des Debuggens müssen Sie einen Haltepunkt setzen, und zwar an der Zeile, die in der `GetItemCount` Funktion `return 5` lautet. Dies kann gemacht werden, indem Sie auf die Zeile klicken und das Symbol `Haltepunkt` in der Werkzeugleiste auswählen oder indem Sie auf den dunklen Rand links von der Zeile klicken. Der Haltepunkt sollte als ein ausgefüllter, roter Kreis erscheinen:

```
#-----------------------------------------------------------
def GetItemCount():
  """This gets the number of items to use in this example"""
  return 5
```

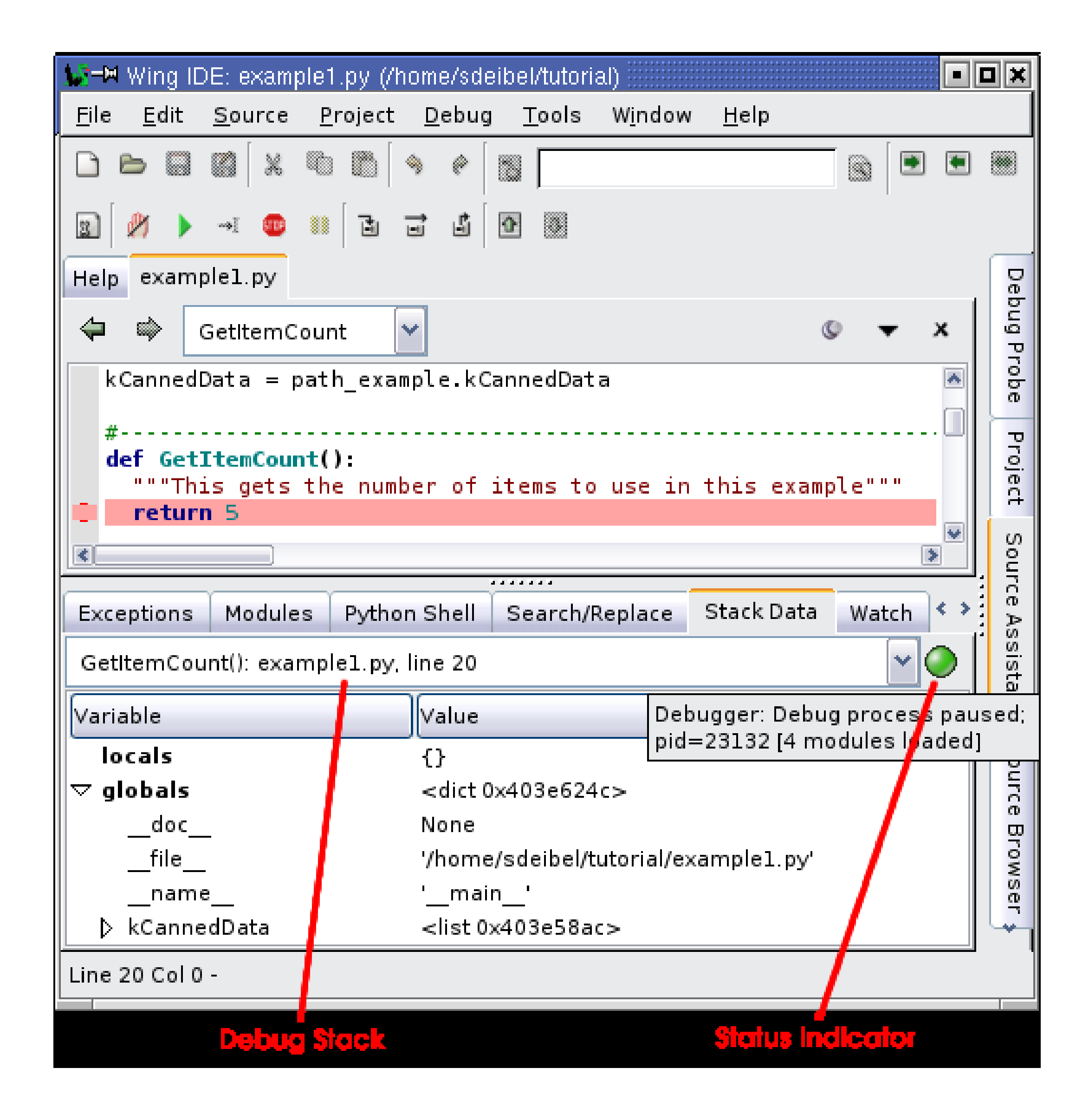
Starten Sie als nächstes den Debugger von der Werkzeugleiste oder dem Menü `Debuggen`. Wing wird den Dialog Debug-Eigenschaften mit den Eigenschaften, die während der Debug-Ausführung verwendet werden, anzeigen. Ignorieren Sie dies jetzt einfach, löschen die Markierung des Kontrollkästchens `Diesen Dialog vor jedem Durchlauf anzeigen` und drücken auf `OK`.

Wing wird bis zum Haltepunkt ausführen, dort anhalten und eine rote Markierung in der Zeile setzen. Sie werden Änderungen in der Werkzeugleiste bemerken; er werden zusätzliche Debug-Werkzeuge angezeigt, wie unten dargestellt:



Ihre Anzeige kann etwas anders aussehen, je nachdem, wie Sie die Einstellungen **Größe der Werkzeugleiste** und **Stil der Werkzeugleiste** konfiguriert haben. Beachten Sie, dass Wing Werkzeughinweie anzeigt, wenn Sie mit der Maus über die Werkzeuge fahren.

Sie können jetzt den Programmstatus an diesem Punkt mit dem **Stack-Daten** Werkzeug prüfen oder indem Sie im Stack-Frame entweder mit den Optionen der Werkzeugleiste oder den Einträgen des Menüs `Debuggen` nach oben oder unten gehen. Das Werkzeug Stack-Daten enthält ein Popup-Menü für die Überprüfung des Programm-Stacks und ein Statuslicht, das den Zustand des Debuggers widerspiegelt. Fahren Sie mit der Maus über diese Anzeige, um eine detaillierte Statusbeschreibung in einem Werkzeugtipp zu sehen.



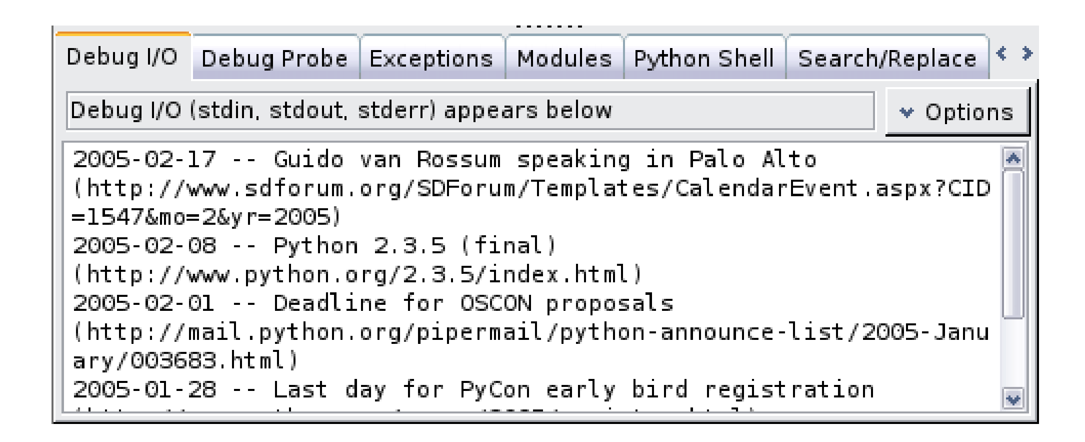
Versuchen Sie als nächstes aus dem eingeschlossenen Aufruf zu `ReadPythonNews` zu

gehen. In diesem bestimmten Kontext können Sie dies mit einem einzigen Klick auf das Symbol `Aus Funktion` in der Werkzeugleiste oder auf den Eintrag des Menüs `Debuggen` erreichen (zwei Klicke auf `In Funktion` funktionieren auch). Diese Funktion eignet sich gut für das Durchschreiten, so dass Sie sich mit den grundlegenden Debugger-Funktionen, die oben beschrieben sind, vertraut machen können.

1.6.1. Tutorial: Debug-I/O

Bevor Sie `ReadPythonNews` verlassen, schlagen Sie das Werkzeug **Debug-I/O** auf, so dass Sie die folgende Ausgabe vom Programm beobachten können. Dies ist auch der Ort, an dem Tastatureingaben stattfinden, wenn der Debug-Code dies erfordert.

Sobald Sie über die Zeile `PrintAsText(news)` schreiten, sollten Sie die folgende Ausgabe sehen:



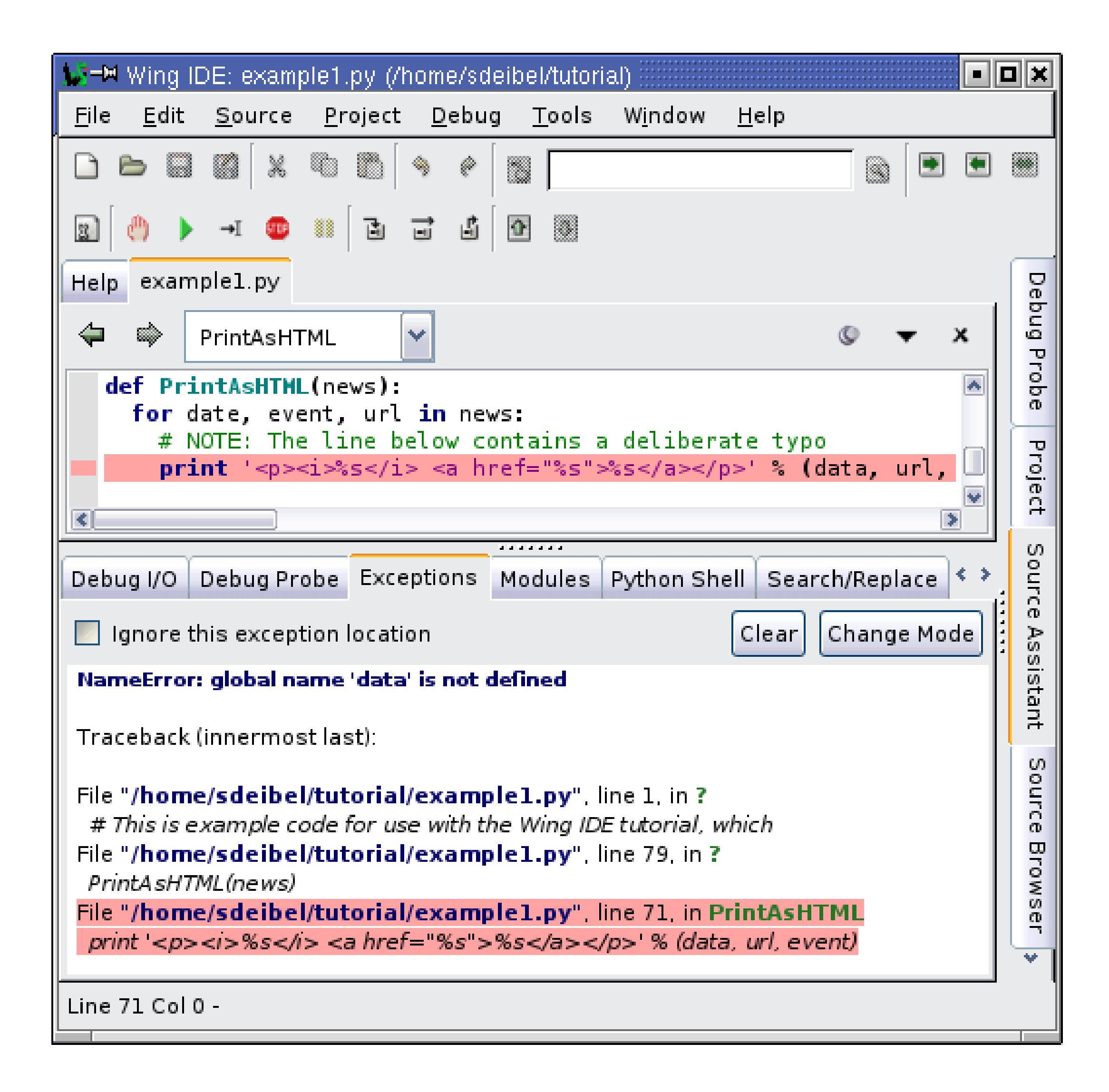
Sie können Wing auch so konfigurieren, dass es eine externe Konsole verwendet. Nutzen Sie dafür das Menü `Optionen` im Werkzeug Debug-I/O. Dies ist für Code hilfreich, der von Einzelheiten der I/O-Umgebung abhängt (wie die Cursor-Steuerung mit speziellen Ausgabezeichen).

1.6.2. Tutorial: Behandlung von Exceptions

Wing's Debugger versucht zu ermitteln, ob eine Exception behandelt (Teil der normalen Operation) oder unbehandelt (führt zu fehlerhafter Programmbeendigung) ist. Dieser Test wird jedesmal durchgeführt, wenn eine Exception im Debug-Prozess angetroffen wird. Wenn eine Exception unbehandelt erscheint, wird Wing sofort an dieser anhalten, so dass Sie den aktuellen Programmstatus, der zu der Exception geführt hat, überprüfen können.

Probieren Sie dies aus, indem Sie die Ausführung des Debug-Prozesses mit dem Symbol `Debuggen` aus der Werkzeugleiste oder dem Eintrag `Debuggen / Fortsetzen` im Menü `Debuggen` fortsetzen.

Wing wird an einer fehlerhaften Code-Zeile in `PrintAsHTML` stoppen und den Fehler im Werkzeug **Exceptions** berichten:



Sie werden bemerken, dass dieses Werkzeug den aktuellen Stack-Frame markiert und dass Sie auf Frames klicken können, um die Rückverfolgung der Exception zu steuern. Immer wenn Sie an einer Exception anhalten, wird das Statuslicht im Werkzeug **Stack-Daten** einen grünen Punkt mit einem Ausrufezeichen anzeigen.

Falsche Positive vermeiden

In einigen Fällen wird Wing fälschlicherweise eine *normale* Exception als unbehandelt markieren und an dieser anhalten. Dies passiert wenn die Exception in C/C++ Erweiterungsmdul-Code verarbeitet wird. Sie können Wing trainieren, diese Exceptions zu ignorieren, indem Sie das Kontrollkästchen `Diese Exception-Position ignorieren` im Werkzeug **Exceptions** anklicken. Wing wird Ihre Wahl speichern und nicht wieder an diesen Exceptions anhalten.

Alternativ können Sie auch die Einstellung **Berichten von Exceptions** verwenden, um den Erkennungsmechanismus für unbehandelte Exceptions auszuschalten. Wählen Sie dafür die Option `Beim Beenden des Debug-Prozesses` aus. Dies funktioniert jedoch nicht gut mit wxPython, PyGTK, extern gestarteten Debug-Prozessen und in einigen anderen Fällen. Den meisten Nutzern empfehlen wir, den Erkennungsmechanismus für unbehandelte Exceptions aktiviert zu lassen.

Zusätzliche Informationen finden Sie im Abschnitt **Exceptions verwalten**.

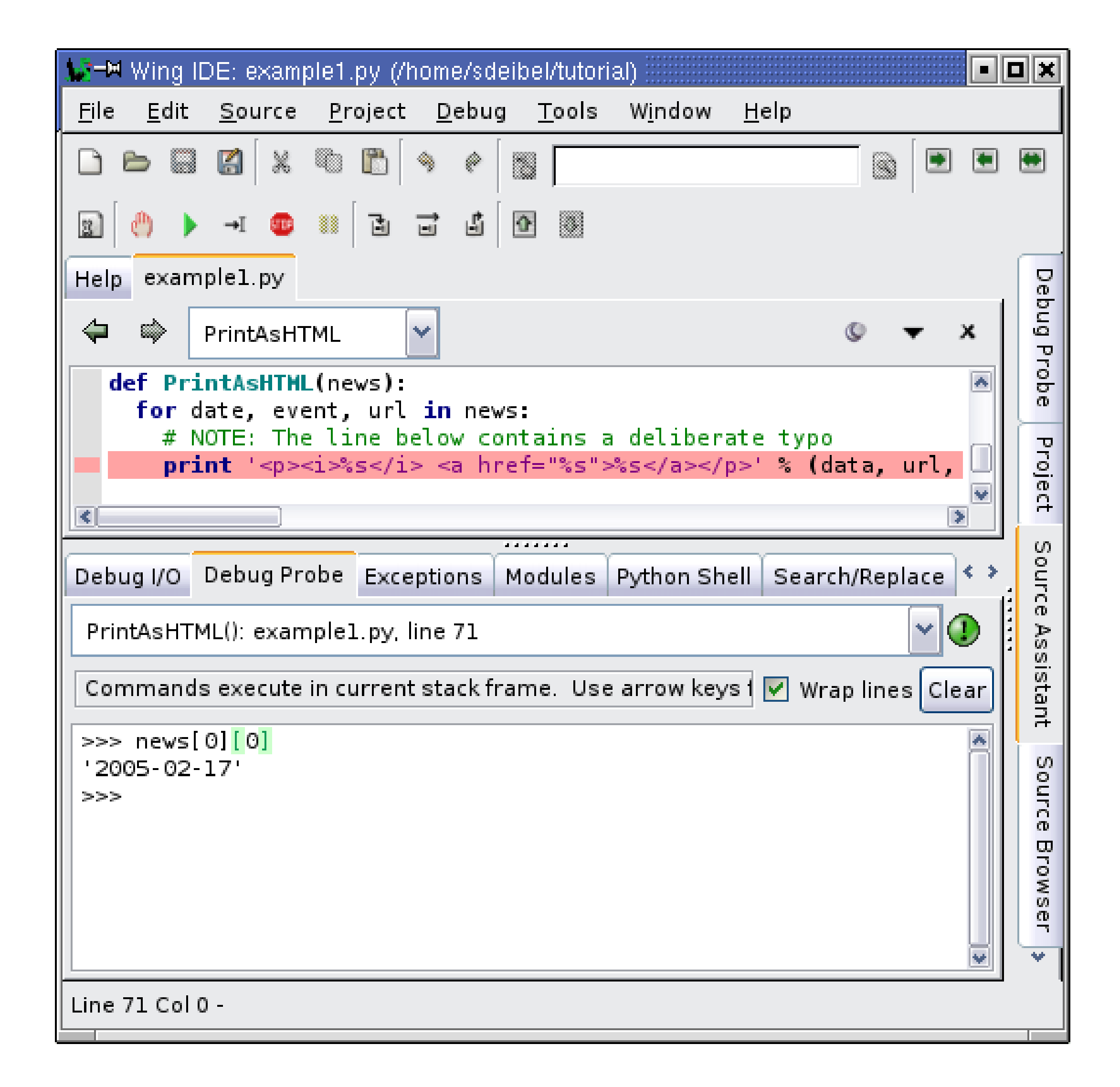
1.6.3. Tutorial: Leistungsfähiges Debuggen mit der Befehlszeile

Wing's **Debug-Test** stellt eine leistungsfähige Möglichkeit zum Suchen und Beheben komplexer Fehler bereit. Dies funktioniert ähnlich wie die Python-Shell, aber es erlaubt die direkte Interaktion mit Ihrem angehaltenen Debug-Programm im Kontext des aktuellen Stack-Frames.

Probieren Sie dies am Punkt der Exception, die wir vorhin erreicht haben, aus, indem Sie dies eingeben:

```
news[0][0]
```

Dies wird das Datum des ersten News-Eintrages drucken:



Probieren Sie als nächstes dies:

```
news[0][0] = ’2004-06-15’
```

Dies ist eine Möglichkeit, den Programmstatus während dem Debuggen zu ändern, was manchmal hilfreich sein kann, wenn Sie Code ausprobieren, der in eine Fehlerbehebung geht. Probieren Sie jetzt dies:

```
PrintAsText(news)
```

Dies führt den Aufruf der Funktion aus und druckt dessen Ausgabe im Debug-Test. Beachten Sie, dass das Werkzeug Debug-I/O nicht für Ein- und Ausgabe verwendet wird, wenn diese aus Befehlen resultieren, die in den Debug-Test eingegeben wurden. Der gesamte Debug-I/O wird zeitweilig hierher umgeleitet.

Hier ist eine andere Möglichkeit. Fügen Sie diesen Code-Block mit Kopieren/Einfügen oder Ziehen und Ablegen in den Debug-Test ein:

```
def PrintAsHTML(news):
  for date, event, url in news:
    print '<p><i>%s</i> <a href="%s">%s</a></p>' % (da-
te, url, event)
```

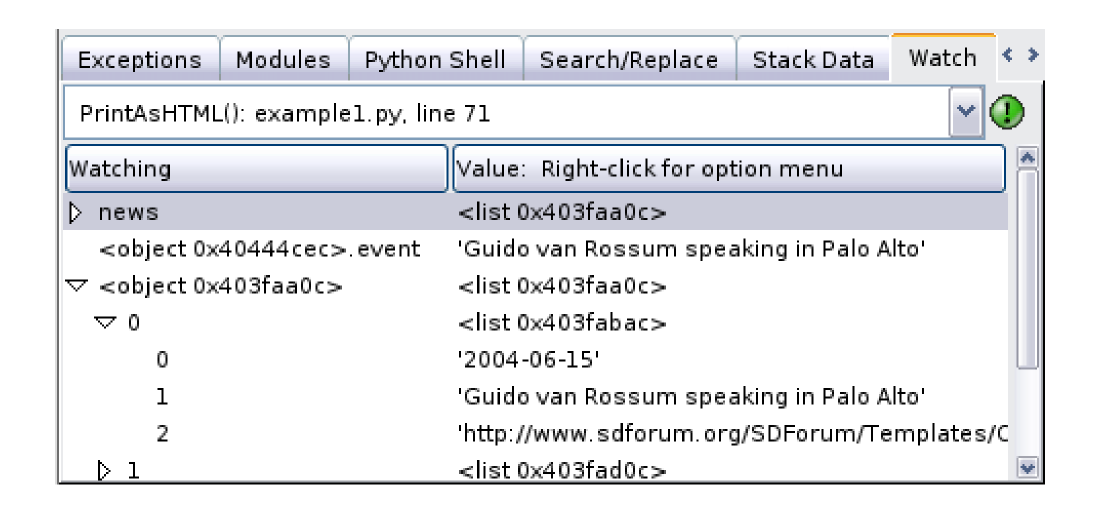
Dies ersetzt die fehlerhafte Definition von `PrintAsHTML` in der `example1.py` Source-Datei, so dass Sie diese nun ohne Fehler wie folgt ausführen können:

```
PrintAsHTML(news)
```

Dies ist bei der Erstellung von Bugfixes hilfreich, wenn die Fehlerbehebung von viel Programmzustand abhängt oder in einem Kontext auftritt, der nur schwer oder sehr zeitaufwendig im Debugger reproduziert werden kann: Schneller interaktiver Test und der Fehler ersetzt mehrere Bearbeiten/Debuggen-Zyklen.

1.6.4. Tutorial: Debug-Daten beobachten

Ein weiteres nützliches Werkzeug bei der Bearbeitung komplexer Fehler ist Wing's Fähigkeit, Debug-Datenwerte auf verschiedene Arten zu beobachten. Dies wird mit dem Werkzeug **Beobachten** vorgenommen:



Werte verfolgen

Während Sie immer noch an der gleichen Exception in `PrintAsHTML` sind, klicken Sie mit der rechten Maustaste auf den `locals` Wert `news` im Werkzeug **Stack-Daten**. Dies präsentiert Ihnen die folgenden Optionen für das Beobachten des Wertes über einen Zeitraum:

- **Nach symbolischem Pfad beobachten** --Dies veranlasst Wing, nach dem symbolischen Namen `news` im aktuellen Stack-Frame zu suchen, wenn Sie debuggen. Wenn Sie diesen Eintrag auswählen, wird im Beobachten-Feld ein Eintrag angezeigt, der folgendermaßen lautet:

  ```
  news                <list 0x40401eec>
  ```

  (die Objekt-ID unterscheidet sich natürlich)

  Dies ermöglicht einen schnellen Zugriff auf Werte, ohne sich dabei durch eine lange locals- oder globals-Liste in der Stack-Datenansicht kämpfen zu müssen.

  Da die Beobachtung über Debug-Sitzungen hinweg sinnvoll ist, wird Sie im Beobachten-Werkzeug gespeichert, bis Sie sie löschen.

- **Nach direktem Verweis beobachten** -- Dies verursacht, dass Wing einen Verweis zu dieser bestimmten Objektinstanz (einer Liste) aufbewahrt. Er wird im Beobachten-Werkzeug angezeigt, solange wie er existiert. Wenn die Verweisanzahl für die Objektinstanz auf null fällt, wird Wing berichten `<Wert nicht gefunden>`.

  Dies ist hilfreich für das Beobachten eines bestimmten Wertes, während Sie durch Teile von Code gehen, die vielleicht keinen Verweis zu ihm halten oder von denen es schwierig ist, die Instanzdaten, auf die verwiesen wurde, zu erreichen.

  Da Objektverweise über Debug-Sitzungen hinweg nicht von Bedeutung sind, werden diese Einträge aus dem Beobachten-Werkzeug entfernt, sobald der Debug-Prozess endet.

- **Nach Parent-Slot beobachten** -- Dies kombiniert die zwei oben genannten Modi, da es einen Verweis zum Parent des gewählten Wertes hält und durch den Unterteil des Wertes nach symbolischen Namen sucht.

  Wenn Sie dies an `event` in `locals` ausprobieren, dann beobachten Sie den Wert `event` innerhalb des bestimmten locals-Dictionaries, anstatt `event` im aktuellen Stack-Frame.

  Diese Technik ist nützlicher, wenn Sie in objektorientiertem Code arbeiten. Dann kann sie dazu verwendet werden, bestimmte Attribute innerhalb einer speziellen Objektinstanz zu beobachten.

  Da der Parent nach Objektreferenz verfolgt wird, werden diese Werte auch aus dem Beobachten-Werkzeug entfernt, sobald der Debug-Prozess endet.

- **Nach Modul-Slot beobachten** -- Diese Option kann verwendet werden, um Werte innerhalb von Modulen zu beobachten, indem das Modul nach Namen in `sys.modules` herausgesucht und der Wert symbolisch verfolgt wird. Dies ist nur verfügbar, wenn Sie mit der rechten Maustaste im Werkzeug Module Werte anklicken, was später im Tutorial diskutiert wird.

  Da nach Modul beobachtete Werte über Debug-Sitzungen hinweg von Bedeutung sind, werden sie im Beobachten-Werkzeug aufbewahrt, bis sie vom Benutzer gelöscht werden.

Ausdrücke beobachten

Es ist auch möglich, den Wert von jedem Python-Ausdruck im Beobachten-Werkzeug zu beobachten. Klicken Sie einfach auf einen leeren Teil der `Beobachten`-Spalte und geben den Ausdruck, den Sie beobachten möchten, ein:

Probieren Sie dies jetzt, während Sie immer noch an der Exception in `PrintAsHTML` angehalten sind. Geben Sie dazu dies ein:

```
news[-1]
```

Dies wird den letzten Eintrag von `news` anzeigen, solange wie einer vorhanden ist, oder `<undefiniert>` oder `<Fehler beim Bewerten>`, wenn der Wert nicht bestimmt werden kann.

Ausdrücke werden im Beobachten-Werkzeug über Debug-Sitzungen hinweg gespeichert, bis sie vom Benutzer entfernt werden.

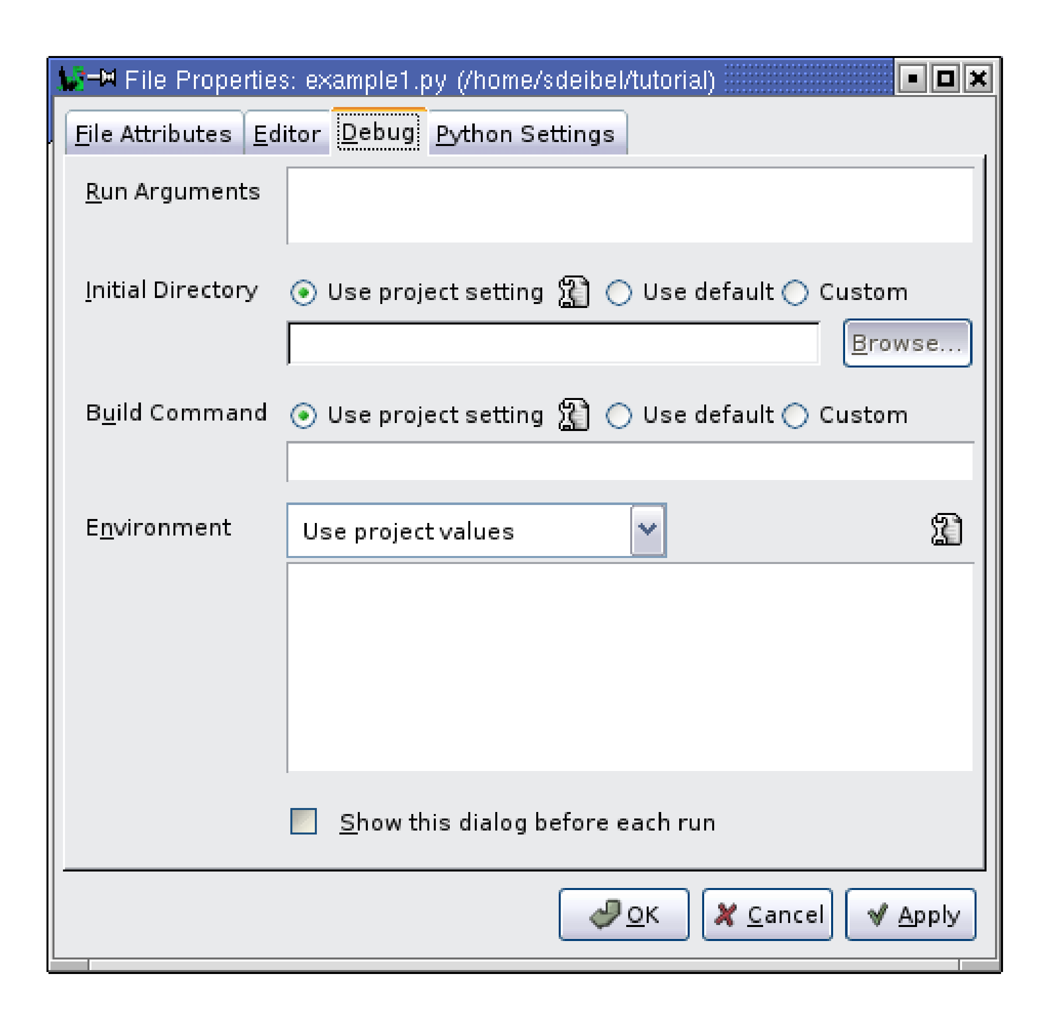
1.6.5. Tutorial: Andere Debugger-Funktionen

Bevor wir zum Rest der IDE-Funktionen fortfahren, hier einige Highlights der weiteren Fähigkeiten des Debuggers, die Sie von Anfang an kennen sollten:

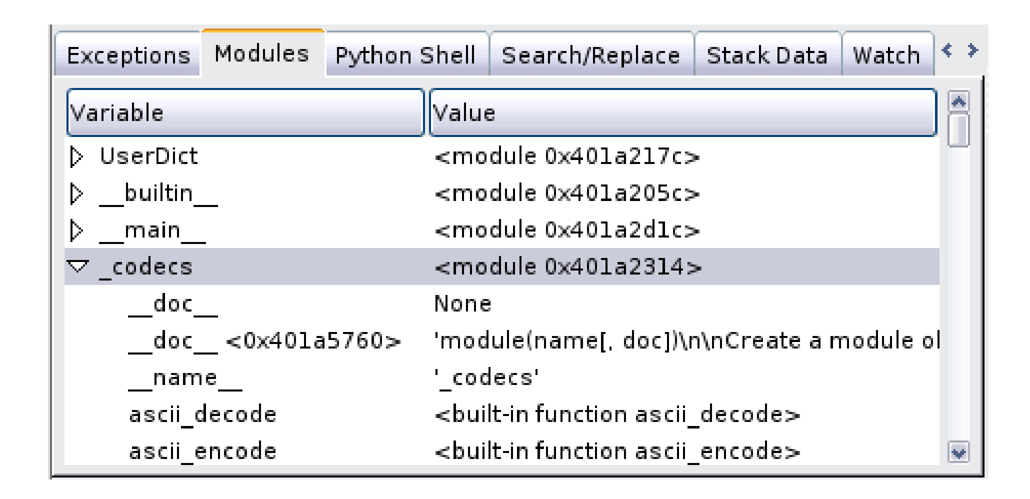
- **Haupt-Debug-Datei** -- Sie können eine Datei in Ihrem Projekt als den Debug-Startpunkt für das Debuggen bestimmen. Wenn dies eingestellt ist, dann wird das Debuggen immer in dieser Datei starten, es sei denn, Sie verwenden den Eintrag `Aktuelle Datei debuggen` aus dem Menü `Debuggen`. Zur Einstellung einer Haupt-Debug-Datei müssen Sie den Eintrag `Aktuelle als Haupt-Debug-Datei einstellen` aus dem Menü `Debuggen` verwenden oder mit der rechten Maustaste auf die Datei im Projekt klicken und `Als Haupt-Debug-Datei einstellen` aus dem Kontextmenü auswählen.

  Die Haupt-Debug-Datei kann mit dem Eintrag `Haupt-Debug-Datei löschen` aus dem Menü `Debuggen` gelöscht werden. Wenn die Einstellung gelöscht ist, dann beginnt das Debuggen immer in der aktuellen Datei des Editors. Welchen Modus Sie verwenden, hängt von der Natur Ihres Projektes ab.

- **Dateieigenschaften** -- Jede Datei in Ihrem Projekt kann Ihre projektweiten Debug-Eigenschaften außer Kraft setzen oder verändern. Dies ist für Projekte hilfreich, bei denen es mehrere Debug-Startpunkte gibt. Die Dateieigenschaften können beim Debuggen auch für die Bestimmung von Argumenten der Befehlszeile verwendet werden. Sie werden über den Eintrag `Aktuelle Dateieigenschaften` im Menü `Source` oder mit `Dateieigenschaften` in den Kontextmenüs des Editors oder Projektes erreicht.



- **Datenansicht der Module** -- Wing filtert standardmäßig Module und einige andere Datentypen aus den Werten, die im Werkzeug Stack-Daten angezeigt werden, heraus. In einigen Fällen ist es nützlich, in Modulen gespeicherte Werte anzuzeigen. Dies kann mit dem Werkzeug **Module** vorgenommen werden, welches eine Liste aller Module ist, die in `sys.modules` gefunden wird:

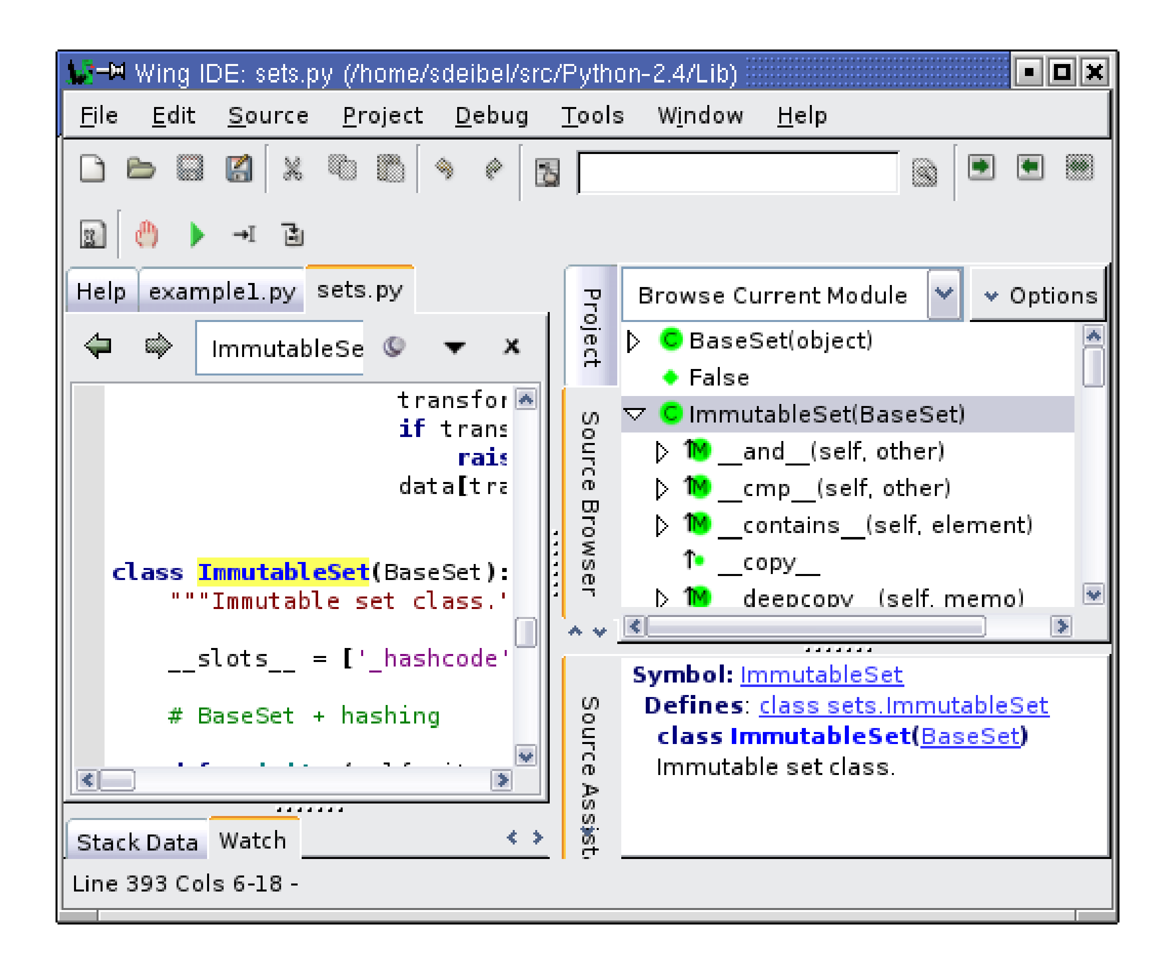


- **Bedingte Haltepunkte** -- Verwenden Sie die Gruppe `Haltepunkt-Optionen` im Menü `Debuggen`, um eine Bedingung zu einem bestehenden Haltepunkt hinzuzufügen oder um einen neuen bedingten Haltepunkt einzufügen. Dies kann sehr nützlich sein, wenn Sie in Ihrem Code vor dem Auftreten eines Fehlers stoppen müssen; Sie können so durch den Code schreiten, der zu dem Fehler führt. Als Bedingung kann jeder beliebige Python-Ausdruck genutzt werden (aber nehmen Sie sich vor Ausdrücken in Acht, die als Nebeneffekt Ihren Programmstatus ändern und bedenken Sie, dass Wing immer an bedingten Haltepunkten stoppen wird, wenn das Bewerten eines Ausdrucks zu einer Exception führt).

- **Remote-Debuggen** -- Wing kann Prozesse debuggen, die unter einem Web-Server als Teil von Zope oder Plone laufen oder die von der Befehlszeile und nicht von Wing gestartet werden. Dies geht über den Rahmen dieses Tutorials hinaus und wird daher im Abschnitt **Extern gestarteten Code debuggen** sowie in den Schnellstart-Anleitungen für **Zope**, **Plone** und **mod_python** beschrieben.

1.7. Tutorial: Source-Browser

Wing IDE Professional enthält einen **Source-Browser**, der zur Prüfung und Navigation der Modul- und Klassenstruktur Ihres Source-Codes verwendet werden kann.



Der Browser wird standardmäßig Klassen, Methoden, Attribute, Funktionen und Variablen, die im gegenwärtig angezeigten Source-Editor (wenn vorhanden) definiert sind, anzeigen. Das Popup-Menü in der oberen linken Ecke des Source-Browsers kann zur Änderung der Anzeige verwendet werden, um alle Klassen oder alle Module im Projekt einzuschließen. Das Optionsmenü (oben rechts) erlaubt die Filterung nach Ursprung, Zugänglichkeit und Art der Source-Symbole. Das Optionsmenü ermöglicht außerdem, dass die Ansicht alphabetisch, nach Typ oder in der Reihenfolge, in der die Symbole in der Source-Datei auftreten, sortiert wird.

Genauso wie bei der Projektanzeige führt ein Doppelklick oder ein rechter Mausklick auf Einträge im Source-Browser dazu, dass diese in einem Editor geöffnet werden. Auch hier erscheint die Option `Auswahl folgen` (im Optionsmenü) und öffnet, wenn aktiviert, vorübergehende Editoren, um für Symbole, die im Source-Browser mit einem einfachen Mausklick oder über die Tastaturnavigation ausgewählt wurden, die Punkte der Definition anzuzeigen.

Der **Source-Assistent** ist im Soure-Browser integriert und wird seinen Inhalt aktualisieren, wenn Sie sich im Baum des Source-Browsers bewegen.

1.8. Tutorial: Suchen

Wing IDE stellt drei verschiedene Oberflächen für das Durchsuchen Ihres Codes bereit. Welche Sie verwenden, hängt von Ihrer Aufgabe ab und sobald Sie alle kennengelernt haben, werden Sie höchstwahrscheinlich mindestens zwei von ihnen nutzen.

Suche mit der Werkzeugleiste

Eine schnelle Möglichkeit, Ihren aktuellen Editor zu durchsuchen, besteht darin, Ihre Suchzeichenkette in den in der Werkzeugleiste bereitgestellten Bereich einzugeben:



Wenn Sie nur Kleinbuchstaben eingeben, wird bei der Suche die Groß- und Kleinschreibung nicht beachtet. Die Eingabe von einem oder mehreren Großbuchstaben führt dazu, dass bei der Suche die Groß- und Kleinschreibung berücksichtigt wird.

Probieren Sie dies jetzt in `example1.py` aus: Tippen Sie `GetItem` in den Suchbereich der Werkzeugleiste ein und Wing wird sofort, ab der Eingabe des ersten Buchstabens, nach übereinstimmendem Text im Editor suchen. Beachten Sie, dass die Betätigung der `Enter` Taste verursacht, dass Wing zum nächsten Treffer weitergeht und, wenn erforderlich, am Ende des Dokumentes zum Anfang der Datei umbricht.

Das Suchen mit der Werkzeugleiste geht in der Datei immer vorwärts (nach unten) und es beginnt an der aktuellen Cursor-Position.

Tastaturgesteuerte Suche

Wenn Sie für eine Suche lieber nicht die Finger von der Tastatur nehmen wollen, dann verwenden Sie die Tastenbefehle, die neben den Einträgen `Mini-Suche` im Menü `Bearbeiten` angegeben sind.

Hiermit können Sie im derzeitigen Editor das Vorwärts- und Rückwärts-Suchen auslösen und optional die aktuelle Auswahl im Editor als Suchzeichenkette verwenden. Sie können außerdem Ersetzen-Operationen starten.

Testen Sie dies in der `example1.py` Datei: Bei Verwendung des Standard-Editor-Modus müssen Sie `Strg-U` drücken. Wenn Sie den Emacs-Modus verwenden, drücken Sie `Strg-S`.

Dies wird im unteren Teil des IDE-Fensters einen Eingabebereich aufschlagen und dieser wird das aktive Feld sein:

Line 18 Cols 4-7 - Search: Get

Fahren Sie mit der Eingabe von `G`, dann `e` und `t` fort. Sie werden bemerken, dass Wing mit jedem Tastendruck vorwärts sucht. Somit müssen Sie nur soviel Text eingeben, wie Sie zum Auffinden des gesuchten Source-Codes benötigen.

Solange der Bereich der Mini-Suche noch aktiv ist, können Sie die gleichen Tastenkombinationen, die Sie für dessen Anzeige genutzt haben, noch einmal verwenden (`Strg-U` oder `Strg-S` in Emacs-Modus) und Wing wird nach der nächsten Übereinstimmung suchen. Wenn kein Treffer gefunden wird, erscheint die Anzeige `Gescheiterte Suche`. Drücken Sie allerdings die Tastenkombinationen der Mini-Suche noch einmal, wird die Suche umgebrochen und startet am Anfang der Datei erneut.

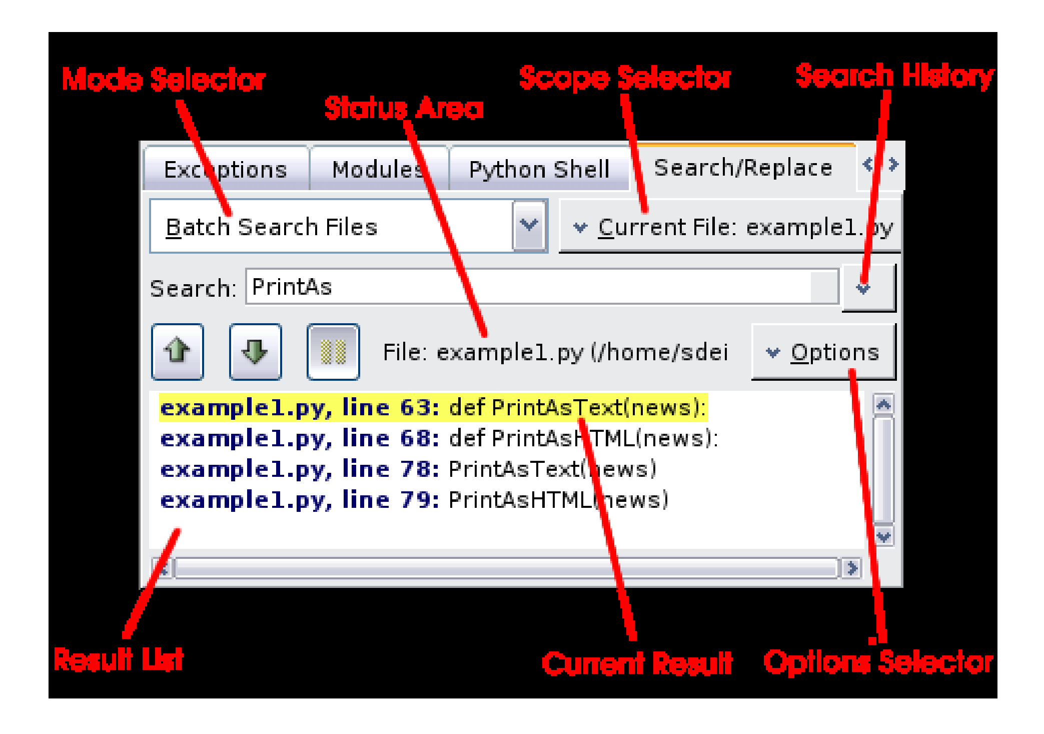
Genauso wie bei Suche mit der Werkzeugleiste führt auch hier die Eingabe von Kleinbuchstaben dazu, dass die Groß- und Kleinschreibung nicht beachtet wird, wohingegen bei der Verwendung von einem oder mehreren Großbuchstaben die Groß- und Kleinschreibung berücksichtigt wird.

Die Suchrichtung kann während dem Suchen durch Betätigung der Tastenkombinationen, die der Mini-Suche vorwärts bzw. rückwärts zugewiesen sind, geändert werden. Sie können die Suche mit der `Esc` Taste oder im Emacs-Modus mit `Strg-G` beenden.

Das tastaturgesteuerte Mini-Ersetzen funktioniert ähnlich, außer dass Ihnen zwei Eingabebereiche bereitgestellt werden, einer für die Suchzeichenkette und einer für die Ersetzen-Zeichenkette. Verwenden Sie `Abfragen/Ersetzen`, damit Sie bei jeder Ersetzen-Position aufgefordert werden, `J` oder `N` einzugeben, oder `Zeichenkette ersetzen`, damit alle Treffer in der Datei global ersetzt werden.

Suchmanager

Das **Suchen/Ersetzen** Werkzeug ist die leistungsfähigste Suchoption, die in Wing IDE zur Verfügung steht. Es unterstützt die Batch-Suche für mehrere Dateien von Ihrem Laufwerk, Ihrem Projekt, von geöffneten Editoren und anderen Sets von Dateien. Es kann die Suche auch mit Wildcards durchführen und ist in der Lage, auf regulären Ausdrücken basierendes Suchen/Ersetzen vorzunehmen.



Bevor wir uns um Einzelheiten kümmern, führen wir eine einfache Batch-Suche in der `example1.py` Datei durch. Wählen Sie `Batch-Suchdateien` in der `Modus-Auswahl` und `Aktuelle Datei` in der `Bereichauswahl` des Suchmanagers aus (dies sind die Voreinstellungen). Geben Sie dann `PrintAs` in den Suchbereich ein.

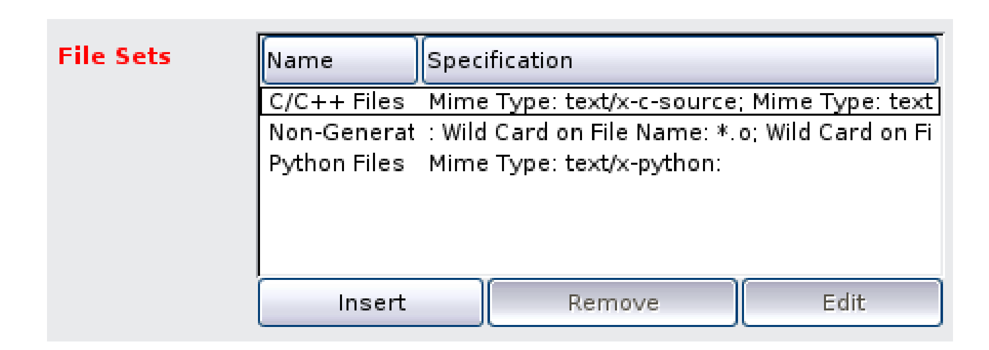
Wing wird sofort mit der Suche beginnen und diese immer dann neu starten, wenn Sie die Suchzeichenkette ändern oder andere Änderungen, welche die Ergebnisse beeinflussen, vornehmen. Wenn Sie damit fertig sind, sollten Sie Ergebnisse wie in dem obigen Screenshot sehen. Klicken Sie auf die erste Ergebniszeile, um diese auszuwählen. Dies wird auch `example1.py` anzeigen, wobei der entsprechende Suchtreffer hervorgehoben ist.

Sie können die vorwärts/rückwärts Pfeile im Suchmanager verwenden, um Ihre Ergebnisse zu durchlaufen. Alternativ können Sie die Einträge `Suche vorwärts` und `Suche rückwärts` aus dem Menü `Bearbeiten` (oder ihre entsprechenden Tastaturkombinationen) nutzen.

Datei-Sets

Verwenden Sie als nächstes die Bereichauswahl, um dort `Alle Dateien im Projekt` auszuwählen, und ändern Ihre Suchzeichenkette auf `HTML`. Dies funktioniert auf die gleiche Weise wie das Durchsuchen einer einzelnen Datei, aber es listet die Ergebnisse für alle Dateien in Ihrem Projekt auf. Sie können auch alle aktuell geöffneten Dateien auf diese Weise durchsuchen.

In den meisten Fällen ist es besser, das Suchen auf ein Teilset der Dateien in Ihrem Projekt zu begrenzen, zum Beispiel nur Python-Dateien. Dies kann mit der Auswahl `Python-Dateien in Ihrem Projekt` vorgenommen werden, aber Sie können mit dem Eintrag `Datei-Sets erstellen/bearbeiten...` in der Bereichauswahl auch andere Datei-Sets definieren. Dies wird die Einstellung Datei-Sets aufschlagen:



Jedes Datei-Set hat einen Namen und eine Liste mit Ein- und Ausschlusskriterien. Jedes dieser Kriterien kann entweder auf den Dateinamen oder auf den MIME-Typ der Datei angewendet werden. Ein einfaches Beispiel wäre es, die Wildcard `*.pas` zu bestimmen, um Übereinstimmungen mit Pascal-Dateien nach Namen zu erhalten, oder die Verwendung des `text/html` MIME-Typen für alle HTML-Dateien.

Suche auf dem Laufwerk

Wing kann direkt auf dem Laufwerk suchen. Probieren Sie dies aus, indem Sie `Batch-Suchlaufwerk` in der Modus-Auswahl bestimmen. Verwenden Sie die Schaltfläche `Durchsuchen...`, die erscheint, um das `tutorial` Verzeichnis, welches Sie vorhin erstellt haben, auszuwählen. Vorausgesetzt Sie haben Ihre Suchzeichenkette nicht geändert, sucht dies in allen Textdateien in diesem Verzeichnis nach `HTML`.

Eine Suche auf dem Laufwerk kann auch rekursiv sein, d.h. in diesem Fall durchsucht Wing auch alle Unterverzeichnisse. Mit der Auswahl von `Rekursiv` aus der Bereichauswahl können Sie dies vornehmen.

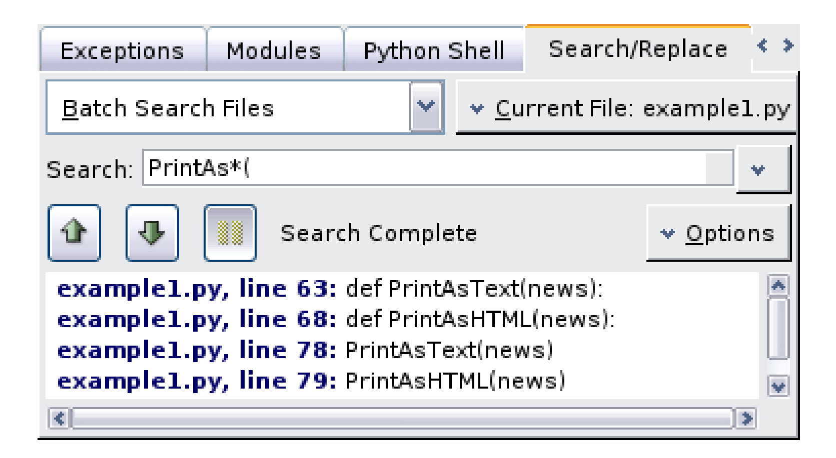
Sie können das Format der Ergebnisliste mit dem Eintrag `Zeilennummern anzeigen` und der Gruppe `Dateiname des Resultats` in der Optionsauswahl ändern. Die Optionsauswahl enthält außerdem viele andere Suchoptionen.

Wildcard-Suche

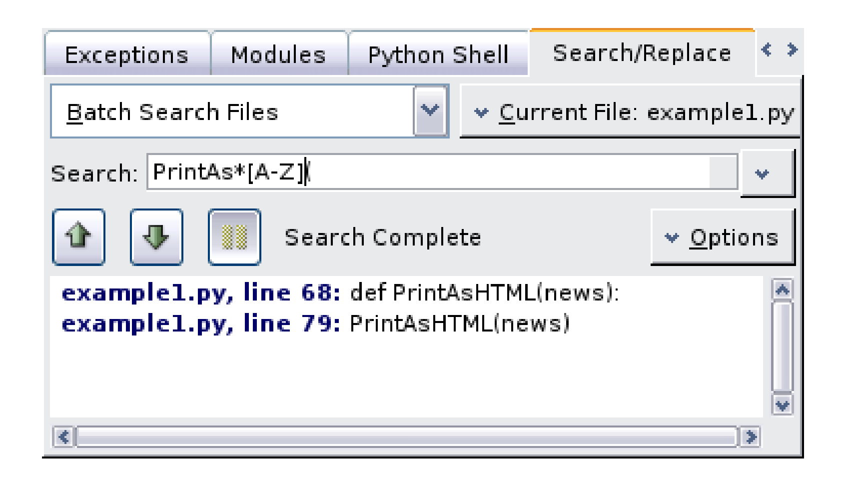
Standardmäßig sucht Wing im Text nach direkten Übereinstimmungen mit den Zeichenketten, die Sie eingeben. Die Wildcard- und reguläre Ausdruckssuche stehen Ihnen aber auch zur Verfügung.

Von diesen beiden ist die Wildcard-Suche leichter zu erlernen. Sie ermöglicht Ihnen, eine Suchzeichenkette zu bestimmen, die `*`, `?` oder Bereiche von Zeichen, die innerhalb von `[` und `]` angegeben werden, enthält. Dies verwendet die gleiche Syntax, wie die, die vom Python `glob` Modul unterstützt wird und sie wird im Benutzerhandbuch auf der Seite **Optionen für Suchen/Ersetzen** detaillierter beschrieben.

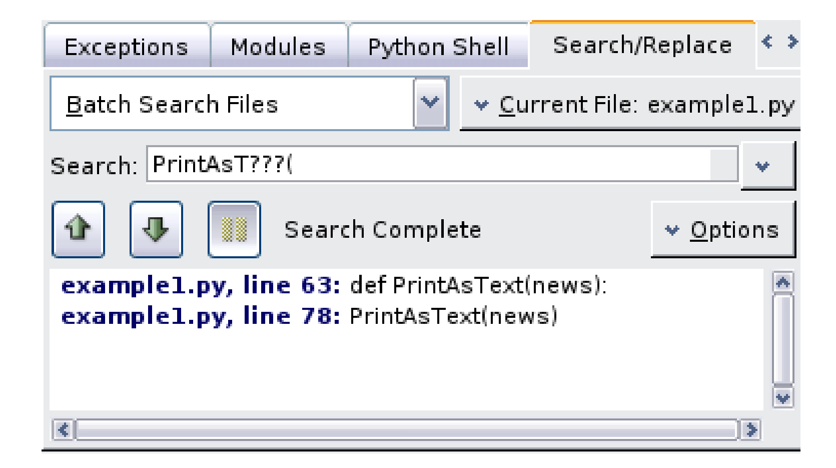
Testen Sie jetzt eine Wildcard-Suche, indem Sie zum Modus `Batch-Suchdateien` zurückkehren und den Bereich auf `example1.py` setzen. Geben Sie als Suchzeichenkette `PrintAs*(` ein. Dies sollte vier Treffer anzeigen, nämlich alle Vorkommen der Zeichenkette `PrintAs`, der null oder mehr Zeichen folgen und die dann das `(` Zeichen enthält:



Versuchen Sie auch, nach `PrintAs*[A-Z](` mit der Option `Groß- und Kleinschreibung` angeschalten, zu suchen. Dies gibt alle Zeichenketten als Treffer an, die mit `PrintAs` beginnen, von null oder mehr Zeichen gefolgt werden, denen dann ein oder mehrere beliebige Großbuchstaben folgen und die dann das `(` Zeichen enthalten:



Probieren Sie schließlich auch `PrintAsT???` aus, welches alle Zeichenketten als Treffer anzeigt, die mit `PrintAsT` beginnen und danach drei beliebige Zeichen enthalten (`?` steht jeweils für ein beliebiges Zeichen):



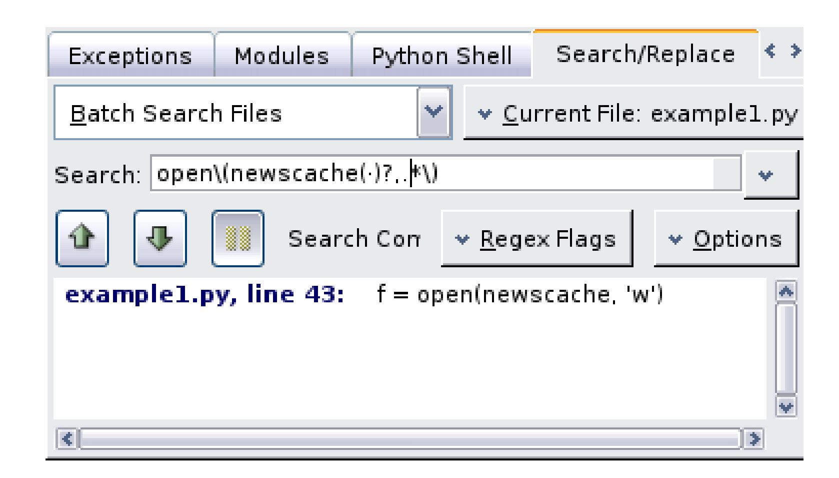
Die Wildcard-Suche kann für das Auffinden von verwandten Source-Symbolen mit nur einer Operation sehr nützlich sein.

Reguläre Ausdruckssuche

Reguläre Ausdrücke können auch für das Suchen verwendet werden. Diese sind für komplizierte Suchaufgaben, wie das Auffinden aller Aufrufe zu einer bestimmten Funktion, die als Teil einer Zuweisungsanweisung auftreten, äußerst hilfreich.

Zum Beispiel zeigt die Zeichenkette `open\(newscache( )?,.*\)` nur die Aufrufe zu der Funktion `open` an, bei denen das erste Argument `newscache` heißt und die mindestens

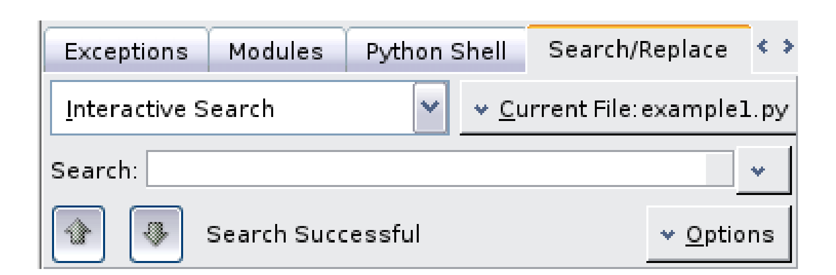
zwei Parameter haben. Wenn Sie dies mit `example1.py` ausprobieren, sollten Sie genau einen Treffer erhalten:



Die Einzelheiten zur Syntax und Verwendung von regulären Ausdrücken können sehr kompliziert sein; daher werden sie in diesem Tutorial nicht behandelt. Lesen Sie dafür die Dokumentation Regular Expression Syntax (Reguläre Ausdruckssyntax) im Python-Handbuch.

Interaktive Suche

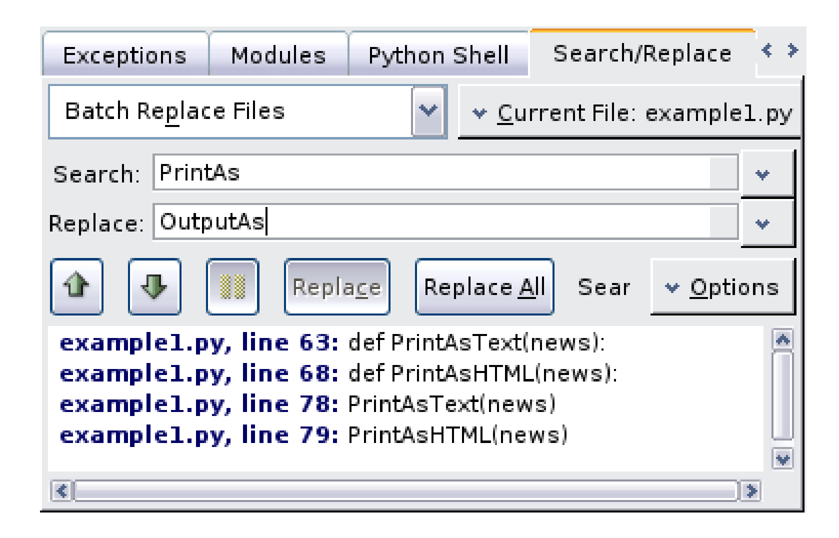
Der Suchmanager kann auch für einfache Suchen verwendet werden, die von der Art her denjenigen ähneln, die mit dem Suchfeld in der Werkzeugleiste bzw. den Werkzeugen der Mini-Suche durchgeführt werden können. In diesem Fall werden die Suchergebnisse nur für einen einzelnen Editor angezeigt und es wird keine Batch-Ergebnisliste zu sehen sein. In diesem Modus sieht der Suchmanager so aus:



Probieren Sie dies mit `example1.py`: Das vorwärts und rückwärts Suchen wird einfach nur die Übereinstimmungen in dem Editor auswählen.

Ersetzen

Wenn einer der Ersetzen-Modi ausgewählt ist, wird Wing einen Bereich für die Eingabe einer Ersetzen-Zeichenkette anzeigen sowie im Suchmanager die Schaltflächen `Ersetzen` und `Alle Ersetzen` hinzufügen:



Versuchen Sie das Ersetzen in der `example1.py` Datei mit `PrintAs` als Suchzeichenkette und `OutputAs` als Ersetzen-Zeichenkette.

Wählen Sie den ersten Treffer in der Ergebnisliste aus und drücken wiederholt auf `Ersetzen`. Es wird jeweils ein Suchtreffer ersetzt. Nach jedem Ersetzen wird die Suche automatisch wieder durchgeführt, außer wenn Sie die Option `Suchen nach Ersetzen` ausschalten. Beachten Sie, dass die Ergebnisliste im Suchmanager periodisch aktualisiert wird, um die vorgenommenen Änderungen widerzuspiegeln. Änderungen können eine nach der anderen im Editor rückgängig gemacht werden. Machen Sie dies jetzt, bis alle vier Suchtreffer wieder in der Ergebnisliste erscheinen.

Versuchen Sie als nächstes `Alle Ersetzen`. Wing wird einfach alle Vorkommen in der Datei auf einmal ersetzen. Wenn diese Option gewählt wird, dann wird ein einziges Rückgängig Machen im Editor den gesamten Ersetzen-Vorgang aufheben.

Ersetzen in mehreren Dateien und auf dem Laufwerk

Wenn Sie im Batch-Modus mit mehreren Dateien im Ergebnisset arbeiten, wird Wing standardmäßig jede geänderte Datei in einem Editor öffnen, egal, ob sie bereits geöffnet ist oder nicht. Dies ermöglicht Ihnen, Änderungen rückgängig zu machen, indem Sie die Dateien nicht speichern oder indem Sie in jedem Editor den Befehl `Rückgängig machen` ausführen.

Im Optionsmenü ist auch ein alternativer Ersetzen-Modus verfügbar. Wenn Sie den Eintrag `Ersetzen arbeitet auf dem Laufwerk` auswählen, wird Wing die Dateien direkt auf dem Laufwerk ändern, anstatt Sie im IDE in Editoren zu öffnen. Diese Methode ist viel schneller, aber wird nicht empfohlen, es sei denn, Sie verfügen über ein Revisionskontrollsystem, das Sie bei Fehlern vor Problemen schützen kann.

Wenn Sie direkt auf dem Laufwerk arbeiten, müssen Sie beachten, dass Wing Änderungen in bereits geöffneten Editoren nur innerhalb des IDEs ersetzen wird. Dies verhindert, dass zwei Versionen einer Datei erstellt werden, wenn bereits Änderungen an der Kopie im IDE vorgenommen wurden. Wir empfehlen, alle Editoren zu schließen, wenn Sie mit dem Modus `Ersetzen arbeitet auf dem Laufwerk` arbeiten oder wählen Sie sofort nach jeder Ersetzen-Operation aus dem Menü Datei die Option `Alle Speichern` aus. Dies verhindert, das Teile eines Ersetzen-Vorganges verloren gehen, was ansonsten zu einer inkonsistenten Anwendung der Ersetzen-Operation auf die Dateien Ihrer Source-Basis führen könnte.

1.9. Tutorial: Source-Assistent mit Klassen

Die früheren Beispiele in diesem Tutorial zu den Aktionen des Source-Assistenten in `example1.py` haben einige seiner Funktionen nicht veranschaulicht, da diese Datei keine Klassen enthält. Wir wollen jetzt noch einmal zu ihm zurückkehren, aber diesmal mit `example2.py` aus Ihrem `tutorial` Verzeichnis. Gehen Sie mit dem Cursor zu der Definition der `end_pre` Methode in `MyHTMLParser` und platzieren ihn dort am Wort `end_pre`. Sie sollten folgendes im Source-Assistenten sehen:

Symbol: end_pre
Defines: method example2.MyHTMLParser.end_pre
def end_pre(self)
Overrides: method htmllib.HTMLParser.end_pre
def end_pre(self)
Overridden to count number of completed <pre></pre> blocks

Der Source-Assistent zeigt auch Informationen über geerbte Klassen an, wenn Sie auf die Klassennamen klicken. Zum Beispiel wird das Klicken auf `self.obj` im Constructor (`__init__()`) von `AnotherClass` dies anzeigen:

Symbol: self.obj
Likely type: instance of example2.MyHTMLParser
class MyHTMLParser(htmllib.HTMLParser)
An example that helps to illustrate some of Wing IDE's features

Hilfe für Wing's Analyser: Beachten Sie die Anweisung `isinstance(obj, MyHTMLParser)` oben von `AnotherClass.__init__()` in `example2.py`: Dies informiert Wing's Source-Analyse-Maschine über den Typ von `obj`. Das Design von Python erschwert umfassende Analysen von objekt-orientiertem Code. Da Typeninformationen aber durch Schlussfolgerungen von anderen Werten erlangt werden können (wie `self.obj` in diesem Fall), können ein paar wenige `isinstance` Hinweise viel dazu beitragen, dass Wing's Fähigkeiten, nützliche Informationen im Auto-Vervollständiger, Source-Assistenten und anderen Werkzeugen anzuzeigen, verbessert werden.

Da Wing's Analyse-Maschine Bedingungen im Code ignoriert, kann in einem Fall, in dem `isinstance` einen kreisförmigen Importfehler zur Programmausführung hinzufügen würde, folgendes verwendet werden:

```
if 0:
  import mymodule
  isinstance(myvalue, mymodule.MyClass)
```

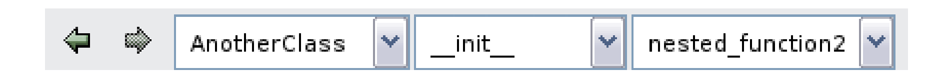
Für Erweiterungsmodule, die in C oder C++ geschrieben sind, kann Wing Python Interface-Dateien, die den Namen des Erweiterungsmoduls plus `.pi` haben, erkennen und verarbeiten. Zum Beispiel kann für ein Erweiterungsmodul, welches als `mymodule` importiert wurde, mit dem Python Skeleton-Code eine Datei für die vom Erweiterungsmodul definierten Funktionen, Attribute, Klassen und Methoden mit dem Namen `mymodule.pi` erstellt werden. Beispiele dieser Dateien finden Sie im Verzeichnis `resources/builtin-pi-files` innerhalb Ihrer Wing IDE Installation.

1.10. Tutorial: Weitere Editor-Funktionen

Im Editor stehen Ihnen eine Vielzahl weiterer Funktionen zur Vefügung, die Sie zumindest kennen sollten:

- **Source-Index** -- Der obere Bereich des Editors enthält eine Reihe von Popup-Menüs, die als Index für Python-Source-Dateien agieren. Wählen Sie aus diesen aus, um durch Ihre Source-Dateien zu navigieren.

Probieren Sie dies aus, indem Sie `example2.py` aus Ihrem `tutorial` Verzeichnis öffnen. Wenn Sie den Cursor an der Zeile `print "nested2"` platzieren, sollten Sie folgendes im Source-Index-Bereich sehen:



Jedes nachfolgende Menü listet die Symbole, die innerhalb des vorhergehenden, verschachtelten Kontexts verfügbar sind, auf.

Wenn Sie die Option `Notizbuchreiter anzeigen` im Optionsmenü des Editors ausgeschaltet haben, wird das Menü der Dateiauswahl folgendermaßen vorangestellt:



- **Gehe zur Definition** -- Es gibt viele Möglichkeiten, zum Punkt der Definition von Symbolen in Ihrem Source-Code zu navigieren. Eine Variante besteht darin, mit der rechten Maustaste auf das Symbol zu klicken und `Gehe zur Definition` auszuwählen. Eine andere Möglichkeit ist, mit dem Cursor zum Symbol zu gehen und aus dem Menü `Source` die Option `Gehe zur gewählten Symboldefinition` auszuwählen (oder F4 drücken). Der Source-Assistent enthält auch Links zum Punkt der Definition.

Versuchen Sie dies in `example2.py` mit einigen Symbolen, die von `htmllib` importiert wurden, wie `HTMLParser` in der Klassendefinition für `MyHTMLParser`. Bedenken Sie, dass die Datei `htmllib.py` im nicht-sticky Modus geöffnet wird und automatisch geschlossen wird, es sei denn, Sie schalten das Stick-Pin-Symbol auf oder bearbeiten die Datei.

Duplicate substitution definition name: „stickpin-stuck“.

- **Gehe zu Zeile** -- Navigieren Sie mit dem `Gehe zu Zeile` Eintrag im Menü `Bearbeiten` schnell zu einer nummerierten Source-Zeile. Im Emacs-Modus wird die Zeilennummer in das Dateneingabefeld im unteren Teil des Fensters eingegeben. Drücken Sie `Enter`, um die Aktion abzuschließen.
- **Tastaturgesteuertes Öffnen von Dateien** -- Testen Sie den `Öffnen mit Tastatur` Eintrag im Menü Datei: Dies zeigt im unteren Teil des IDE Fensters eine interaktive Dateiauswahl an, die für das Öffnen von Dateien sehr viel schneller ist, als die Verwendung des Standard-Dialogs für die Dateiauswahl, und die Ihnen ermöglicht, eine Dateiauswahl zu treffen, ohne dabei die Hände von der Tastatur zu nehmen. Verwenden Sie `Esc` zum Abbrechen oder `Enter` zur Auswahl einer Datei sowie die Pfeiltasten, um die Auto-Vervollständigungsliste, die beim Tippen aufgeschlagen wird, zu durchsuchen.
- **Automatische Einrückung** -- Während Sie tippen, rückt Wing die Zeilen entsprechend der statischen Analyse Ihres Codes automatisch ein. Dies kann mit der Einstellung **Automatisch einrücken** deaktiviert werden.

Eine andere Weise, in der Wing die Code-Analyse verwendet, ist bei der automatischen Einrückung während Sie tippen und beim Ändern der Einrückung oder dem Umbrechen von Code. Wenn Sie zum Beispiel einen Code-Block auswählen und die Tab-Taste drücken, wird der gesamte Block entsprechend der korrekten Position der ersten Zeile relativ zur vorhergehenden nicht-leeren Code-Zeile neu eingerückt. Die Option `Text ausrichten` im Source-Menü verwendet auch den Source-Analyser, um das neue Umbrechen einer einzelnen logischen Zeile von Python-Code zu erzwingen.

- **Block-Einrückung** -- Die `Tab`-Taste ist so definiert, dass sie die aktuelle Zeile oder Blöcke von Zeilen einrückt, anstatt ein Tab-Zeichen einzufügen (was mit `Strg-Tab` gemacht werden kann). Die Einrückung von einer oder mehreren ausgewählten Zeilen kann vergrößert oder reduziert werden, indem Sie die Einrückungsgruppe in der Werkzeugleiste verwenden, welche für diesen Zweck die folgenden Symbole enthält:



Einzelne Zeilen oder ganze Blöcke können auch automatisch zu ihrer entsprechenden Position eingerückt werden; die richtige Position wird durch eine Analyse der vorhergehenden Zeile bestimmt. Wird ein Bereich von Zeilen ausgewählt, dann wird der gesamte Block ein- oder ausgerückt, ohne dabei die relativen Einrückungen innerhalb des Blocks zu verändern. Dies wird mit dem folgenden Werkzeugleistensymbol gemacht:



Die Einrückungsfunktionen stehen auch im Menü `Source` zur Verfügung, wo außerdem Ihre Tastaturbefehle aufgelistet sind.

- **Block-Kommentierung** -- Einheiten von Code können mit dem Menü `Source` schnell auskommentiert werden bzw. können die Kommentare für die Bereiche schnell aufgehoben werden.
- **Klammernsuche** -- Wing hebt während Sie tippen übereinstimmende Klammern hervor, außer wenn Sie die Einstellung **Automatische Klammernsuche** deaktiviert haben. Der Eintrag `Klammernsuche` im `Source` Menü führt dazu, dass Wing den gesamten Code auswählt, der in den am nächsten gelegenen übereinstimmenden Klammern, von der aktuellen Einfügungsposition im Editor ausgehend, enthalten ist. Die wiederholte Anwendung dieses Befehls wird die Datei nach außen gehend und vorwärts durchlaufen.

- **Neuformatierung von Text** -- Code kann mit der Option `Text ausrichten` im Menü `Source` neu formatiert werden. Dies begrenzt das Umbrechen auf eine einzelne, logische Code-Zeile, so dass es für das Umbrechen einer Argumentliste, einer langen Liste oder eines Tuple verwendet werden kann, ohne dabei den umgebenden Code zu verändern.

- **Konvertierung von Einrückungsstilen** -- Wing's Werkzeug **Einrückung** kann verwendet werden, um den Stil der in Source-Dateien gefundenen Einrückungen zu analysieren und umzuwandeln. Siehe Abschnitt **Einrückungsmanager** für Einzelheiten.

- **Falten** -- Wing erlaubt das Falten von Editor-Code, um Bereiche, die zur Zeit nicht von Interesse sind, zu verstecken. Sie können dieses Verhalten deaktivieren, indem Sie die Einstellung **Falten aktivieren** ausschalten. Das Falten erfolgt nur visuell, so dass eine Auswahl und das Kopieren über ein Falte hinweg, den Text einschließlich der versteckten Teile kopieren wird. Das Falten kann nützlich sein, um eine schnelle Zusammenfassung von den Inhalten einer Source-Datei zu erhalten. Zum Beispiel wird die Anwendung der Option `Alle Zusammenklappen` gefolgt von `Aktuelle Mehr Erweitern` auf eine Klasse eine Liste ihrer Methoden anzeigen. Einzelheiten finden Sie im Abschnitt **Falten** im Benutzerhandbuch.

- **Makros** -- Tastatur-/Befehlsmakros sind verfügbar. Siehe Abschnitt **Tastaturmakros** im Benutzerhandbuch für Einzelheiten.

1.11. Tutorial: Weiterführende Quellen

Herzlichen Glückwunsch! Sie haben das Tutorial beendet. Bei der Arbeit mit Wing IDE an Ihrem eigenen Software-Entwicklungsprojekt können die folgenden Ressourcen hilfreich sein:

- Wing IDE Support-Website
- **Wing IDE Benutzerhandbuch**
- **OS X Schnellstart-Anleitung**
- **Zope Schnellstart-Anleitung**
- **Plone Schnellstart-Anleitung**
- **wxPython Schnellstart-Anleitung**

- **PyQt Schnellstart-Anleitung**
- **mod_python Schnellstart-Anleitung**

Migration von Wing IDE 1.x

Die Umstellung von Wing IDE 1.x auf Wing IDE 2.0 sollte relativ schmerzlos sein. Wenn Sie Wing IDE 2.0 das erste Mal starten, wird es Ihre Einstellungen von Wing IDE 1.x automatisch übernehmen und in Ihr **Verzeichnis der Benutzereinstellungen** platzieren (beachten Sie, dass sich der Ort dieses Verzeichnisses geändert hat).

- **Lizenzierung**

Wing IDE 2.0 hat einen neuen Lizenzmanager, der Wing IDE 1.1 Lizenzen nicht verwenden kann. Probelizenzen werden jetzt direkt vom Produkt erworben und alle Lizenzen müssen aktiviert werden (an einen bestimmten Host gebunden), bevor sie verwendet werden können. Zusätzliche Informationen finden Sie im Abschnitt **Lizenzen**.

Wing IDE 1.1 Lizenzen können in unserem Online-Shop aufgerüstet werden. -- Dies ist kostenlos, wenn die Lizenz am oder nach dem 28. August 2003 erworben wurde. Wurde die Lizenz vor dem 28. August 2003 erworben, erhalten Sie das Upgrade zum halben Preis.

- **Umwandlung von Projekten**

Wing IDE 1.x Projektdateien werden umgewandelt, indem sie geöffnet und als unbenannt markiert werden, so dass sie unter einem neuen Namen gespeichert werden können. Sie sollten Ihre alten Projektdateien nicht überschreiben, wenn Sie beabsichtigen, Wing IDE 1.x noch zu verwenden. Wing IDE 2.0 Projektdateien können in Wing IDE 1.x nicht gelesen werden.

- **Anpassung der Benutzeroberfläche**

Wing IDE 2.0 führt eine fast vollständig umgestaltete Benutzeroberfläche ein. Während die grundlegende Bedienung der Anwendung von der Tastatur her ähnlich geblieben ist, hat sich das Aussehen der Oberfläche stark verändert und bietet viel mehr Optionen für die persönliche Anpassung. Lesen Sie das Kapitel **Anpassung** im Benutzerhandbuch, um zu erfahren, wie Sie Ihre Oberfläche neu konfigurieren.

Diejenigen, die den alten, aus mehreren Fenstern bestehenden Ansatz von Wing IDE 1.1 bevorzugen, sollten einen Blick auf die Einstellung **Fensteraufteilung** werfen. Außerdem können Sie selbst im Modus des kombinierten Fensters beliebig viele Dokumentfenster anlegen sowie Werkzeuge und Editoren aus den Notizbuchfeldern in separate Fenster und zwischen bestehenden Fenstern verschieben. Ihre Konfiguration wird in Ihrer Projektdatei gespeichert.

- **Neue Funktionen**

Dies sind die wichtigsten neuen Funktionen, die mit Wing IDE 2.0 eingeführt werden:

Neue Benutzeroberfläche -- Neu gestaltet für einen besseren Arbeitsablauf und mehr Flexibilität. Die neue Benutzeroberfläche (basierend auf Gtk2) sieht außerdem viel ansprechender aus und ist äußerst anpassungsfähig.

Source-Assistent -- Dieses Werkzeug fügt eine kontext-entsprechende Anzeige der Call-Signatur, des Doc-Strings, dem wahrscheinlichen Datentyp und vielem mehr hinzu. Diese neue Funktion schließt sich der Auto-Vervollständigung, dem Source-Index, der Gehezu-Definition und dem Source-Browser an, die zusammen die leistungsfähigen Code-Intelligenzwerkzeuge von Wing IDE bilden.

Neu gestalteter Suchmanager -- Er unterstützt jetzt das Suchen und Ersetzen für viele Dateien, Wildcards und reguläre Ausdrücke. Der neue Suchmanager ist außerdem bei der Erstellung und dem Testen von regulären Ausdrücken hilfreich. *Integrierte Hilfe* -- Die gesamte Wing IDE Dokumentation ist jetzt direkt innerhalb des IDEs verfügbar. Sie erreichen sie über das Hilfemenü und in vielen Fällen über die Werkzeug-Tipps, die angezeigt werden, wenn die Maus über verschiedene Bereiche der Benutzeroberfläche bewegt wird.

Integrierter Debug-I/O -- Der I/O des Debug-Prozesses erscheint im Werkzeug Debug-I/O innerhalb des IDEs, obwohl die Verwendung einer externen Konsole auch immer noch eine Option darstellt.

Deutsche Lokalisierung -- Deutsche Versionen der Benutzeroberfläche und der Dokumentation stehen jetzt zur Verfügung.

Modul-Datenansicht -- Durchsuchen Sie Programmdaten, die mit `sys.modules` starten.

Beobachtungsmanager -- Dies kombiniert die alten Funktionen Expression Evaluator und Wertverfolgung in einem leistungsfähigen Werkzeug. Klicken Sie auf das Werkzeug, um Ausdrücke für die Bewertung hinzuzufügen oder verfolgen Sie Werte nach Objektverweis oder symbolischen Namen, indem Sie mit der rechten Maustaste auf eine beliebige Debug-Datenansicht klicken. *Neu gestaltete Source-Kontextmenüs* -- Diese erscheinen oben im Editorbereich als eine Reihe von Menüs, eines für jeden verschachtelten Bereich am aktuellen Einfügungspunkt, anstatt als ein einzelnes riesiges hierarchisches Menü.

Optionale Editor-Reiter -- Wählen Sie zwischen Dateien unter Verwendung der Notizbuchreiter oder eines Popup-Menüs oben im Editor aus.

Integrierte Fehlerberichterstattung und Feedback -- Verwenden Sie die Einträge des Hilfemenüs, um direkt vom Produkt aus Fehler zu berichten oder Feedback einzureichen.

Neuer Lizenzmanager -- Er vereinfacht das Erhalten von Probelizenzen und die Installation von erworbenen Lizenzen. Außerdem fügt er die Lizenzaktivierung hinzu, um Unternehmen bei der Überwachung der Lizenzverwendung zu unterstützen und um die leichtfertige gemeinsame Nutzung von Lizenzen (welche leider unsere zukünftige Produktentwicklung gefährdet) zu reduzieren.

Nachrichtenmanager -- Erstellt ein Protokoll von Statusnachrichten nach Untersystemen.

Was spricht für die Verwendung von Wing IDE?

Wing IDE ist eine leistungsfähige Software-Entwicklungsumgebung für Python-Programmierer. Da es die Zeit des Code Schreibens und des Debuggens reduziert, können Sie mit Wing qualitativen Code viel schneller entwickeln.

> **„The best Python IDE currently available.“** *(Das beste Python IDE, das zur Zeit auf dem Markt ist.)*
>
> *-- Stephen Scherer, Ed.D., Jan 2004*
>
> **„A real power tool to use for the development and debugging of complex Python applications“** *(Ein echtes Power-Tool für die Entwicklung und das Debuggen von komplexen Python-Anwendungen.)*
>
> *-- InformIT, Sep 2001*

- **Schnelleres, genaueres Coden** -- Wing bietet Ihnen an den Kontext angepasste Code-Vervollständigungsoptionen und Dokumentation, erlaubt Ihnen die schnelle Navigation zum Punkt der Definition und markiert syntaktische Fehler, wenn Sie Code eingeben. Durch die Fülle von tastaturgesteuerten Such- und Bearbeitungsfunktionen, einschließlich einem Emacs-Modus, liegt jede Menge der Coding-Leistung in Ihren Fingerspitzen.
- **Schnelleres Debuggen** -- Wing findet Fehler sehr schnell, denn es erkennt schwere Exceptions und stoppt sofort an diesen. Sie haben so die Möglichkeit, durch Ihren Code zu schreiten und ihn im Kontext des Problems zu überprüfen. Bedingte Haltepunkte, Wertverfolgung nach symbolischen Verweis und Objektverweis sowie die Interaktion an der Befehlszeile mit Ihrem angehaltenen Debug-Programm beschleunigen das Eingrenzen eines Fehlers und das Entwerfen einer Lösung für ihn. Flexible Startoptionen und Remote-Debuggen erlauben Ihnen, mit Ihrem Code im Kontext zu arbeiten, selbst wenn Sie unter einem Web-Server, Zope oder Plone, einer eingebetteten Skriptsprache oder von der Befehlszeile gestartet ausführen.

- **Schnelleres Lernen des Codes** -- Wing lässt Sie unbekannten Code schnell verstehen, denn es verwendet seine leistungsfähigen Suchmöglichkeiten, seinen Source-Code-Browser, seine Editor-Navigationsmenüs, die Gehe-zur-Definition sowie die Fähigkeit des Debuggers, Programme in Aktion zu beobachten.

Mehr Produktinformationen finden Sie auf unserer Website unter www.wingware.com.